Schrifterkennung in Videobildern

Diplomarbeit

im

Fachbereich Photoingenieurwesen und Medientechnik

an der Fachhochschule Köln

Vorgelegt von

Axel Näther

Mat.-Nr. 11017774

Email: axel.naether@gmx.de

Referent: Prof. Dr.-Ing. Klaus Ruelberg

Korreferent: Prof. Dr. rer. nat. Dietmar Kunz

Köln, im Mai 2004

Character recognition in video images

Thesis

at the Department

of

Imaging Sciences and Media Technology

University of Applied Sciences Cologne

Presented by

Axel Näther

Mat.-Nr. 11017774

Email: axel.naether@gmx.de

First Reviewer: Prof. Dr.-Ing. Klaus Ruelberg

Second Reviewer: Prof. Dr. rer. nat. Dietmar Kunz

Cologne, May 2004

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung und Zielsetzung

Fernseh- und Videobilder in Fernsehsendungen oder in digitalen Videoarchiven, aber beispielsweise auch auf Grafiken basierende Inhalte von Internetseiten enthalten eine große Anzahl von in sie eingebetteten Texten. Die Lokalisierung und Erkennung dieser Texte eröffnet eine Vielzahl von Anwendungsmöglichkeiten.
So können Texte beispielsweise als Ergebnis der Suche und Verarbeitung von großen digitalen Videoarchiven mit Inhalten von Nachrichtensendungen oder Dokumentationen zum Erstellen von Datenbanken dienen [07]. Da in Nachrichtensendungen sehr oft Einblendungen von Orten, Namen, Uhrzeiten oder sonstigen Informationen vorhanden sind, können diese Grundlage für Datenbanken werden, durch die sehr schnell auf einzelne Beiträge zugegriffen werden kann. Auch dienen diese Datenbanken der Verknüpfung unterschiedlicher Inhalte von Beiträgen miteinander. Für Firmen die im Fernsehen Werbung schalten ist es von Interesse zu überprüfen, ob und gegebenenfalls wann der eigene Werbespot gesendet wurde. Auch hier wäre eine automatische Erkennung von Schriftzügen wie beispielsweise des Produktnamens und die gleichzeitige Aufzeichnung der Uhrzeit und des Sendekanals von Interesse [04]. Eine weitere Anwendung ist die Erkennung von Inhalten in Webseiten welche nicht textbasiert sind, also zum Beispiel in Flash oder ähnlichen Formaten vorliegen. Auf die in diesen Seiten hinterlegten Informationen können Standard-Suchmaschinen nicht zugreifen, weshalb die Informationen bei einer Suchanfrage nicht berücksichtigt werden [06].
An diese Problem und die Anwendungsmöglichkeiten anknüpfend, ist es das Ziel die-

ser Arbeit Möglichkeiten zur Erkennung von in Fernsehbildern eingebetteten Schriften, wie beispielsweise "www.schrift-im-videobild.de" aufzuzeigen und als reinen ASCII-Text zu extrahieren. Dazu wurde eine Sammlung von verschiedenen Testbildern erstellt, die unterschiedliche von den Fernsehanstalten eingeblendete Texte enthalten. Darüber hinaus soll ein Programm mit Hilfe der Programmiersprache C++ erstellt werden, das die Testvorlagen für eine Texterkennung durch kommerzielle Optical Character Recognition (OCR) Software, wie sie für die Erkennung von reinen Binärbildern (zum Beispiel eingescannte Schreibmaschinentexte) genutzt wird, vorverarbeitet. Um dieses zu erreichen wurden Möglichkeiten untersucht, die im Bild enthaltenen Schriften zu detektieren, zu segmentieren und in einem weiteren Verarbeitungsschritt in ein neues Bild zu extrahieren, wobei das schlussendliche Bildresultat nach Möglichkeit nur noch schwarze Schrift vor weißem Hintergrund enthalten soll.

Kapitel 2

Grundlagen

2.1 Schriftcharakteristika

Generell muss zwischen zwei verschiedenen Arten von eingebetteten Texten unterschieden werden:
Zum Einen kann ein Text als Teil der Szene bei der Aufzeichnung vorhanden sein. Es handelt sich hierbei um sogenannten "scene text", der beispielsweise unter anderem als Schriftzug auf Produkten, Text auf Schildern oder auch als Werbelogo im Bild bei der Übertragung zu sehen ist.
Zum Anderen werden eine Vielzahl von Texten, genannt "artificial text", erst in der Postproduction in die Sendungen eingesetzt [09]. Beispiele hierfür wären Untertitel, Programmhinweise, Namen von Moderatoren und Interviewgästen und dergleichen. "Scene text" erscheint sehr oft zufällig und ungeplant unter unterschiedlichsten Beleuchtungsverhältnissen, Winkeln, Größen und Ausrichtungen in den Videobildern. Dagegen wird der "artificial text" sehr gezielt verwendet. Die Textausrichtung ist fast immer horizontal und hat immer einen hohen Kontrast zur Umgebung um gut lesbar sein zu können. Da "artificial text" von größerem Interesse, als der "scene text" bei der Auswertung von Informationen in Videobildern ist, werden sich in den folgenden Kapiteln die Auswertungen und Angaben auf diesen Bereich beschränken. Dabei lassen sich für den "artificial text" in den Videobildern einige Gemeinsamkeiten festhalten, die in Tabelle 2.1 überblicksweise dargestellt sind [09].

- Die Texte stehen im Vordergrund eines Videobildes und werden daher nicht von anderen Inhalten überdeckt.
- Die einzelnen Buchstaben der Texte sind meistens einfarbig und ohne Textur.
- Die Buchstaben unterliegen gewissen Beschränkungen in der Größe; ein einzelner Buchstabe wird nicht den gesamten Bildschirm füllen oder so klein werden, dass er vom Betrachter nicht lesbar ist.
- Der Kontrast zwischen Text und Hintergrund ist ausreichend groß um ein leichtes Lesen zu gewährleisten.
- Die Texte erscheinen entweder statisch, also an einer festen Stelle im Bild oder sie haben eine feste Bewegungsrichtung, nämlich horizontale oder vertikale Laufschriften.
- Die Texte erscheinen immer in mehreren Frames. Bei Lauftexten wie beispielsweise einem Filmabspann, haben diese wenn mehrere einzelne Bilder nacheinander betrachtet werden zum Teil einen durch die Bewegung verursachten vertikalen oder horizontalen Versatz innerhalb aufeinanderfolgender Einzelbilder.
- Buchstaben erscheinen in Gruppen (Worten) mit horizontaler Ausrichtung, ähnlich den Zeilen in Schriftdokumenten. Für asiatische Länder müßte die Ausrichtung der Worte in vertikaler Richtung berücksichtigt werden.
- Einzeln im Bild stehende Buchstaben oder Ziffern sind in der Regel nicht vorhanden oder nur in Kombination mit anderen Wörtern und Ziffernfolgen zu finden.

Tabelle 2.1: Schriftcharakteristika in Videobildern

2.2 Erstellung der Testvorlagen

Bei der analogen elektronischen Fernsehbildübertragung wird das Bild zeilenweise, sequentiell übertragen. Die Auflösung der übertragenen Fernsehbilder ist abhängig von der Anzahl der Zeilen und der Anzahl der Bildpunkte je Zeile. In der europäischen Fernsehnorm wurde eine Bildwechselzahl von 25 Bildern/s (aufgeteilt in 50 Halbbilder/s) mit 625 Zeilen je Bild festgelegt. Davon sind bei analoger Übertragung 575 Zeilen in vertikaler Richtung in jedem Fernsehbild sichtbar.
Die Testvorlagen wurden in SDTV-Qualität[1] am PC mit einer Videokarte und mit Hilfe der Software "DScaler"[2] erstellt. Dabei wurden bewußt Vorlagen aus unterschiedlichen Sendungen und von unterschiedlichen Sendern ausgesucht.
Ein RGB-Bild besteht aus den Kanälen Rot, Grün und Blau (RGB). Jeder der Kanäle repräsentiert ein zweidimensionales Feld mit $n * m$ Bildpunkten. Die Software "Dscaler" liefert Bilder im BMP oder Tiff Format in der Größe $720 * 576$ Pixel, wobei die Auflösung 8 Bit je Kanal beträgt. Die von der Software unterstützten Filter wie beispielsweise Rauschreduktion oder Scharfzeichnen wurden nicht aktiviert um unveränderte Bilddaten zu erhalten.
Alternativ zu der Erstellung und Verarbeitung von einzelnen Bildern ist zum Beispiel die Verarbeitung mittels einer Video- oder Frame-Grabber-Karte denkbar. Hiermit wären TV-Sendungen direkt im zu entwickelnden Programm einlesbar. Die Verarbeitung einzelner Testvorlagen hat jedoch den Vorteil, dass einzelne Verarbeitungsschritte des Programms beliebig oft wiederholbar sind. Die Testvorlagen sind im Windows Bitmap Format (BMP) gespeichert. Device Independent Bitmaps (DIB) sind unter Windows ein Standard um Rasterbilder geräteunabhängig zu speichern. DIB können mit unterschiedlichen Farbtiefen vorliegen. Die Testvorlagen haben eine Farbtiefe von 8 Bit je Kanal, wodurch sich insgesamt $256^3 \approx 16,7$ Millionen Farben darstellen lassen. DIB bestehen aus den für die Programmierung wichtigen folgenden Teilen: Einem Bitmapfileheader, einem Bitmapinfoheader, einer Farbtabelle und den eigentlichen Bilddaten.
Der Bitmapfileheader kennzeichnet die Datei eindeutig als Bitmap. Er enthält unter

[1] SDTV = Standard Definition Television.
[2] Download: www.dscaler.org

anderem den ASCII-Code des Strings "BM", die Dateigröße und den Offset zum Anfang der Bilddaten.
Der darauffolgende Bitmapinfoheader enthält unter anderem Informationen über die Breite und Höhe des Bitmaps, welche bei den Testvorlagen $720 * 576$ Pixel betragen sollte, den Typ einer möglichen Datenkompression (JPEG-Komprimierung für Bitmaps wird ab Windows 98 unterstützt), die Farbtiefe in Bits pro Pixel (BPP) und die Größe der Bilddaten in Byte. Je nach Typ des Bitmaps ist den eigentlichen Bilddaten außerdem eine Farbtabelle vorangestellt. Die einzelnen Bildpunkte enthalten in diesem Fall keine diskreten Werte für R, G und B, sondern nur einen Verweis auf die interne Farbtabelle, wodurch Speicherplatz gespart wird. Bei den unbearbeiteten Testvorlagen ist keine Farbtabelle vorangestellt. Je Pixel sind 3 Byte Bilddaten nötig, je ein Byte für den R,G und B Wert. Einen Überblick über einen Teil der Bitmapfileheader-Struktur gibt die Tabelle 2.2. Die Tabelle 2.3 gibt einen Überblick über die Bitmap-Farbtabelle.
Für die Verarbeitung mittels der Kantenfilter werden die Testbilder in Grauwertbilder konvertiert. Bei farbigen Bitmap-Vorlagen werden die Farben aus drei Grundfarben additiv zusammengemischt. Bei den Testvorlagen lassen sich je Grundfarbe (RGB) $2^8 = 256$ Farben darstellen. Insgesamt lassen sich 256^3, also ca. 16 Millionen Farben darstellen. Sollen Grauwerte dargestellt werden, so unterscheidet man im Allgemeinen 256 Graustufen mit Werten zwischen 0 und 255, wobei hier 0 für Schwarz und 255 für Weiß steht. Damit benötigt man je Grauwert genau 1 Byte.
Bei der Umwandlung von RGB-Werten in Grauwerte kann man die RGB-Werte als Koordinaten von Vektoren auffassen, die einen dreidimensionalen Vektorraum aufspannen; den RGB-Farbraum. Auf den drei Achsen des Raums werden die Grundfarben RGB mit Werten zwischen 0 und 255 aufgetragen. Die Grauwerte, also Werte bei denen Rot Grün und Blau denselben Zahlenwert aufweisen, liegen auf der Diagonalen des Farbraumes. Dies entspricht dem Vektor vom Nullpunkt (0,0,0) zum Punkt (255,255,255). Um für eine beliebige Farbe einen entsprechenden Grauwert zu berechnen kann man die Projektion des Farbvektors $\vec{F}$ auf die Richtung des Grauwertvektors $\vec{G}$ (der Raumdiagonalen) berechnen:

$$|\vec{F}|_{\vec{G}} = \frac{\vec{F} * \vec{G}}{|\vec{G}|} . \tag{2.1}$$

Name	Offset vom Dateianfang (hex.)	Größe in Byte	Bedeutung
bfType	h00	2	Typ der Datei. Für ein BMP muss dies der ASCII Code des Strings "BM" h4D42 sein.
bfSize	h02	4	Größe der Datei in Byte.
bfReserved1	h06	2	Reservierter Eintrag, muß 0 sein.
bfReserved2	h08	2	Reservierter Eintrag, muß 0 sein.
bfOffBits	h0A	4	Offset vom Dateianfang zum Beginn der Bilddaten.
biWidth	hH4	4	Breite des Bitmaps in Pixeln.
biHeight	hH8	4	Höhe des Bitmaps in Pixeln.
biBitCount	hHE	2	Farbtiefe des Bitmaps in Bits pro Pixel (bpp).
biXPels	h14	4	Horizontale Auflösung des Zielgeräts in Pixeln pro Meter.
biYPels	h18	4	Vertikale Auflösung des Zielgeräts in Pixeln pro Meter.

Tabelle 2.2: Bitmapfileheader-Struktur

Name	Offset vom Dateianfang (hex.)	Groeße in Byte	Bedeutung
rgbBlue	h36	1	Gibt die Intensität der Farbe Blau an (0 bis 255).
rgbGreen	h37	1	Gibt die Intensität der Farbe Grün an (0 bis 255).
rgbRed	h38	1	Gibt die Intensität der Farbe Rot an (0 bis 255).
rgbReserved	h39	1	Reservierter Eintrag, muss 0 sein.

Tabelle 2.3: Aufbau der Bitmap-Farbtabelle

Die Normierung des Grauwertvektors $\vec{G}$ auf 255 Längeneinheiten ergibt:

$$\frac{|\vec{F}|_{\vec{G}}}{|\vec{G}|} * 255 = \text{gesuchter Grauwert}\,. \tag{2.2}$$

Gemäß der Annahme, dass R, G und B jeweils gleich große Werte annehmen können (hier jeweils max. 255) und dementsprechend auch 256 Grauwerte (0 bis 255) dargestellt werden sollen, lassen sich die beiden oberen Formeln stark vereinfachen zu:

$$(R + G + B)/3 = \text{gesuchter Grauwert}\,. \tag{2.3}$$

2.3 OCR - Optical Character Recognition

OCR oder auch "optische Zeichenerkennung" steht für einen Softwaretyp, der eingescannte, als Bilddateien abgespeicherte Vorlagen auf enthaltene Schriftzeichen untersucht und diese als ASCII-Schriftsätze abspeichert. Damit werden die in den Bilddaten enthaltenen Schriften mit einem gewöhnlichen Editor wie zum Beispiel Microsoft Word oder Star-Writer bearbeitbar und müssen nicht manuell vom Nutzer eingegeben werden.

Die Verarbeitung der Vorlagen geschieht durch die OCR-Software in mehreren Schritten. Einige OCR-Programme unterstützen neben der Erkennung von Schwarzweiß-Vorlagen, wie beispielsweise Schreibmaschinentexten oder Formularen auch farbige Vorlagen und Handschriftenerkennung. Je nach Software werden die Bilddaten, sofern sie einer Farbvorlage entstammen, in ein Graustufen-Bild umgewandelt. Danach wird die Vorlage durch die OCR-Software analysiert, wobei die Software, die in der Vorlage vorhandene Schrift zu finden und in einzelne Bereiche aufzuteilen versucht. Dies können Absätze, Zeilen oder einzelne Wörter sein. Danach erfolgt die Auswertung in den einzelnen Teilen und die Ausgabe der gefundenen Zeichen in eine Textdatei. Hierbei besteht oftmals die Option bereits vorhandene Textformatierungen aus der Vorlage zu übernehmen. Zusätzlich bieten einige Programme die Option einer Filterung mittels Wörterbuch und mit Hilfe von Sprachregeln. Dazu muss der Nutzer vor der Verarbeitung der Vorlagen eine Sprache festlegen.

Für die Texterkennung existieren mehrere Methoden:

- **Pattern Matching:** Pattern Matching oder auch Musterabgleich führt eine einfache Kongruenzprüfung zwischen den segmentierten Buchstaben der Vor-

lage und den in der OCR-Software hinterlegten Zeichensätzen durch. Hierfür muss in der OCR-Software eine große Anzahl verschiedener Muster -also Schriften- in diversen Skalierungen vorliegen.

- **Feature Recognition:** Feature Recognition analysiert die segmentierten Zeichen nach typischen Merkmalen, wie Linien, Kreisformen sowie Winkelungen der Linien und Formen zueinander, wobei für jeden Buchstaben die typischen Eigenarten in der OCR-Software abgespeichert sind.

- **Feature Extraction:** Diese Methode ist eine Weiterentwicklung der Feature Recognition-Methode. Die geometrischen Merkmale werden erkannt und in einzelne Gruppen zusammengefaßt, so wird beispielsweise berücksichtigt, dass einige Buchstabenverbindungen wie etwa 'ie' besonders häufig vorkommen.

Pattern Matching, Feature Recognition und Extraction werden von den meisten OCR-Programmen kombiniert. Daneben existieren noch weitere Verarbeitungsmöglichkeiten durch die OCR-Software; interne Wörterbücher in unterschiedlichen Sprachen können Fehler bei der Erkennung reduzieren, auch sind zum Teil grammatikalische Prüfungen ganzer Sätze möglich zudem sind viele Programme "lernfähig", das heißt, dass der Anwender vorgeben kann wie nicht erkannte Zeichen verarbeitet werden sollen beziehungsweise inwieweit die Verarbeitung automatisch und ohne Rückfragen an den Nutzer seitens der Software geschieht.

2.3.1 Einflussfaktoren auf die Texterkennungsrate

- **Vorlagenqualität:** Die Vorlagenqualität wird durch unterschiedliche Faktoren bestimmt. Ist der Anwender von OCR-Software in Besitz einer Vorlage in Papierform, so kann diese mit ausreichender Qualität gescannt werden. Bei den aus den Fernsehbildern gewonnenen Testvorlagen ist die Auflösung durch die Sendeform bestimmt. Eine Verbesserung der Auflösung könnte hier beispielsweise durch Sendungen im HDTV-Format[3] erreicht werden. Störeffekte auf dem Übertragungsweg sind auch nicht auszuschließen und können ebenfalls die Qualität beeinflussen. Eine optimale Vorlage für die Texterkennung

[3]HDTV = High Definition Television

stellt zum Beispiel eine eingescannte mit Schreibmaschine beschriebene oder gedruckte Seite dar, also ein Binärbild mit zwei Farben: Schwarz für den Text und Weiß für den Hintergrund.

- **Handschrift:** Die Erkennung von handschriftlichen Vorlagen ist für OCR-Programme besonders schwierig, da die Handschrift personengebundene Variationen aufweist und einzelne Buchstaben schwer zu separieren sind.

- **Farbige Vorlagen:** Farbige Vorlagen werden meist intern in ein Graustufenbild umgewandelt, wobei insbesondere Text in vollfarbigen Flächen oftmals nicht oder nur unzureichend erkannt wird.

- **Ähnliche Zeichen:** Die Zeichen I, i, 1, 0 (Null) und O (Buchstabe O) sind schwer voneinander zu unterscheiden und führen zu Fehlern in der Erkennung.

- **Ligaturen:** Ligaturen sind zusammengezogene Buchstaben die als ein Buchstabe erkannt werden, beispielsweise werden manchmal die Buchstaben r und n als Buchstabe m erkannt.

- **Skalierung und Ausrichtung:** Die Skalierung und Ausrichtung der Texte haben einen entscheidenden Einfluß auf die OCR. In gewöhnlichen Dokumenten sind die Schriftarten in Bezug auf die Seitengröße vergleichsweise klein. Im Gegensatz dazu können bei Fernsehbildern die Schriften vergleichsweise groß werden, so zum Beispiel bei Einblendungen von Werbeschriften.

Dies sind nur einige wichtige Punkte, die für die OCR relevant sind. Im Hinblick auf die OCR in Fernsehbildern sind diese Punkte jedoch von Bedeutung und werden im Weiteren zu berücksichtigen sein.

2.3.2 Verwendete Software für die OCR

OCR-Software wird von einer Vielzahl von Herstellern angeboten. Zum einem gibt es für den Endkunden Standard-Software mit meist ähnlichen Benutzeroberflächen und fast immer gleichen Verarbeitungsschritten: Scannen einer Vorlage, Texterkennung, Ausgabe des Texts in verschiedenen Dateiformaten. Je nach verwendeter Software

lassen sich zusätzlich unterschiedliche manuelle Einstellungen vornehmen.
Zum anderen gibt es Software-Bibliotheken und Entwicklungsumgebungen für Programmierer mit deren Hilfe sich eigene OCR-Programme und Anwendungen gestalten lassen. Insbesondere sei hier die Firma ScanSoft genannt, die eine "OCR-Engine" vertreibt. Beispielhaft für verschiedene OCR-Software seien hier drei verschiedene Programme erwähnt:

- Abbyy FineReader 6.0 Corporate Edition
- TextBridge Pro 11
- OmniPage Pro 12.0

Der Umgang mit den Programmen ist sehr schnell erlernbar und die Oberflächen sind nahezu selbsterklärend. Hier einige Features der Software die in den meisten Punkten von allen eben genannten Programmen unterstützt werden [10]:

- Hervorragende Erkennungsgenauigkeit und Layout-Übernahme
- Einfache Bedienung
- Öffnen/Speichern von PDF-Dokumenten
- Mehrsprachige Erkennung
- Speichern von Ergebnissen in mehreren Dateiformaten
- WYSIWYG Text-Editor
- XML Output und Integration in Microsoft Office Word 2003
- Barcode-Erkennung
- Unterstützung von Multifunktions-Peripheriegeräten im Netzwerk
- Anwendung zum Ausfüllen von Formularen

- Verteilte Dokumentverarbeitung für die Gruppenarbeit
- Benutzerdefinierte Wörterbücher und Sprachen
- Juristische und medizinische Wörterbücher

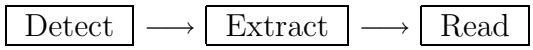
2.3.3 Verarbeitung der Testvorlagen mit kommerzieller OCR-Software

Da die Standard OCR-Software auch Farbvorlagen verarbeiten kann, bietet sich der Versuch an die erstellten Testvorlagen direkt in der Software zu verarbeiten.
Bei diesem Versuch ist sehr schnell festzustellen, dass die Software hierfür nicht konzipiert ist beziehungsweise sehr unterschiedliche und unbefriedigende Ergebnisse bei der Suche und der Erkennung der enthaltenen Texte liefert. Beispielsweise findet das Programm OmniPagePro in der Testvorlage Nr. 27 (siehe Anhang) problemlos den Text "www.o2online.de/business" und den Text "pure business", während das Programm AbbyyFineReader bei der Verarbeitung eine Warnung ausgibt, nach der das Dokument keinerlei Text enthalte. Insgesamt kommen unterschiedliche Programme zu sehr uneinheitlichen Ergebnissen. Aber auch bei mehrmaliger Verarbeitung derselben Vorlage in immer demselben Programm schwanken die Ergebnisse stark. Insgesamt ist die Erfolgsquote bei der Schrifterkennung in den Testvorlagen sehr gering und sicherlich nicht ausreichend.

2.4 Grundlagen der OCR in Videobildern

Bezüglich der OCR in Videobildern sind hauptsächlich im englischsprachigen Raum, einige Publikationen erschienen, die sich mit der OCR in digitalen Videobildern und ähnlichen Datenformaten beschäftigen.
In diesem Bereich werden für die Video-OCR meistens ähnliche grundlegende Verarbeitungsschritte mit zum Teil sehr unterschiedlichen Algorithmen vorgenommen. In [13] gehen Toshio Sato, Takeo Kanade und andere Autoren speziell auf die Anforderungen der Video-OCR von Untertiteln für die Indizierung von digitalen News-Libraries ein, wobei insbesondere Probleme bei der Erkennung von kleinen Schriftar-

ten und sehr komplexen Text-Hintergründen thematisiert werden. Huiping Li, Omid Kia und David Doermann beschreiben in [14] die Möglichkeiten der Bildverbesserung in digitalen Videobildern. In [06] werden von Daniel Lopresti und Jiangying Zhou Versuche zum Erkennen von Texten in WWW-Bildern zusammengefaßt. Vor allem die Aspekte der Layoutanalyse und die Auswirkungen des Anti-Aliasing von Schriften in WWW-Bildern auf die OCR sind hier zu erwähnen.
Allgemein kann eine Einteilung der Video-OCR in drei wesentliche Verarbeitungsschritte vorgenommen werden:



"Detect" steht für das Auffinden des Textes innerhalb eines Videobildes beziehungsweise innerhalb einer Reihe von mehreren Videobildern (vergleiche 3.6). Hierbei sind wie bereits unter 2.1 beschrieben, insbesondere die Eigenschaften der Texte in Videobildern von Nutzen. "Extract" beschreibt das Extrahieren der Texte aus dem Umfeld. Ziel einer Verarbeitung der Videobilder sind Bilddaten die lediglich noch den Text enthalten und bei denen der gesamte Hintergrund einfarbig (weiß) erscheint. "Read" steht für das Erkennen der gefundenen Texte. Dies kann durch Standard OCR-Software geschehen.
Diese drei Schritte der Verarbeitung lassen sich durch diverse Teilschritte ergänzen, mit dem Ziel die Texterkennungsrate innerhalb der Videobilder zu optimieren.

Kapitel 3

Texterkennung in den Testvorlagen

Ein Teilgebiet der digitalen Bildverarbeitung ist die automatische Lokalisierung und Segmentierung. Hierbei werden Objekte in einem Bild mit Hilfe von bestimmten Merkmalen detektiert und anschließend vom Hintergrund beziehungsweise einzelne Objekte voneinander getrennt. Allgemein unterscheidet man zwischen flächenorientierten und konturorientierten Verfahren. Flächenorientierte Verfahren trennen Gebiete mit unterschiedlichen Grauwerten, Kontrasten oder Farben voneinander. Dagegen nutzen konturorientierte Verfahren den Gradienten der Grauwerte um die Objektkonturen zu extrahieren. Im Folgenden soll ein Überblick über mögliche Verfahren gegeben werden, wobei insbesondere jene Verfahren von Interesse sind, welche die unter 2.1 genannten textspezifischen Eigenarten berücksichtigen.

3.1 Lokalisierung der Textbereiche mittels Kantendetektion

Wie in 2.1 beschrieben sind Texte in Bildern durch unterschiedliche Merkmale charakterisiert. In der Literatur werden sie teilweise als Gebiete von hohem Kontrast beschrieben. Bildregionen mit Text enthalten vor allem eine große Anzahl von Ecken und Kanten. Auf diesen Merkmalen beruhen viele Algorithmen zur Lokalisierung von Texten in (Video-)Bildern. Neben dem Auffinden der Textgebiete müssen aber auch

Verfahren zum Extrahieren der Texte aus dem Hintergrund gefunden werden.
Kanten sind Diskontinuitäten im Verlauf der (Grauwert-)Intensitäten. Das Auffinden von Kanten in Bildern ist abhängig von unterschiedlichen Einflussfaktoren: der Qualität des Bildes (Rauschen, Auflösung), der verwendeten Algorithmen und der Art der Kanten sowie deren Orientierung innerhalb des Bildes. Da es sich im Ortsfrequenzraum betrachtet bei Kanten um hochfrequente Bildanteile handelt und auch Rauschen eine hochfrequente Störung im Bild darstellt, sollte in Abhängigkeit von dem verwendeten Operator vor der eigentlichen Kantendetektion eine Filterung durchgeführt werden, um ein "Verschmieren" der Kanten zu vermeiden. Dies könnte zu einer Verschlechterung der Lokalisation führen.
Je nach verwendetem Operator werden in den Bildern bevorzugt Kanten in horizontaler und vertikaler Richtung gefunden, daneben gibt es aber auch richtungsunabhängige (isotrope) Operatoren. Im Folgenden werden einige bekannte Kantenoperatoren aufgeführt und beispielhaft ist unter Abbildung 3.2 bis 3.5 eine Testvorlage dargestellt, die mit den Operatoren verarbeitet wurde.

3.1.1 Gradientenverfahren

Für die Detektion von Kanten sind Filteroperationen geeignet, welche die Grauwertveränderungen verstärken und Bildbereiche mit konstantem Grauwert unterdrücken. An einer Kante hat die erste Ableitung der Kantenfunktion ein Extremum, die zweite Ableitung hat an der Stelle der größten Kantensteigung beziehungsweise des größten Kantenabfalls einen Nulldurchgang [16]. Eine Vielzahl von Kantenoperatoren berechnen die Kanten mittels Gradienten. Der Gradient einer stetigen Funktion zweier Variablen ist durch

$$(\partial f(x,y)/\partial x,\, \partial f(x,y)/\partial y) \tag{3.1}$$

definiert. Der Betrag des Gradienten ist ein Maß für die Steilheit einer Kante. Die Position der Kante befindet sich an der Stelle des maximalen Betrages des Gradienten. Für den Betrag gilt:

$$l = \sqrt{(\partial f(x,y)/\partial x)^2 + (\partial f(x,y)/\partial y)^2}\,. \tag{3.2}$$

Für die Richtung gilt:

$$\Theta = arctan(\frac{\partial f(x,y)}{\partial x} / \frac{\partial f(x,y)}{\partial y}) \,. \tag{3.3}$$

3.1.2 Sobel-Operator

Eine sehr häufig verwendete Bildverarbeitungsoperation ist die diskrete Faltung oder diskrete Filterung. Die Faltung einer Bildinformation $B(m,n)$ mit einer diskreten Filterfunktion $H(i,j)$ ist beschrieben durch:

$$B'_{m,n} = \sum_{i=-I}^{I} \sum_{j=-J}^{J} H_{m,n} * B(m-i, n-j) \,. \tag{3.4}$$

Die Koeffizienten der Filterfunktion $H(i,j)$ werden auch als Filtermaske bezeichnet. Werden für die Elemente in der Maske H auch negative Koeffizienten zugelassen, so ergeben sich Differenzoperatoren. In der Regel ist die Maske deutlich kleiner als das betrachtete Bild. Für die Kantendetektion sind unterschiedliche Filtermasken H gebräuchlich [16].

Der Sobel-Operator detektiert bevorzugt horizontale und vertikale Kanten [17]. Die Filtermasken des Sobel-Operators sind gegeben durch:

$$H_x = \begin{vmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{vmatrix} \qquad H_y = \begin{vmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{vmatrix} . \tag{3.5}$$

Die Werte für die neuen Bildpunkte $B'_{x,y}$ ergeben sich aus:

$$I_x = B_{x,y} * H_x \qquad I_y = B_{x,y} * H_y \tag{3.6}$$

$$B'_{x,y} = \sqrt{I_x^2 + I_y^2} \,. \tag{3.7}$$

3.1.3 Laplace-Operator

Der Laplace-Operator ist ein richtungsunabhängiger Operator das heißt es werden Kanten in allen Richtungen gleich gut erkannt. Die punktsymmetrische Maske H ist gegeben durch:

$$H = \begin{vmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{vmatrix} . \tag{3.8}$$

Alternativ zu 3.8 sind in der Literatur aber auch folgende Filterkerne zu finden, die alle ähnliche Eigenschaften besitzen [19]:

$$H_1 = \begin{vmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{vmatrix} \quad H_2 = \begin{vmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{vmatrix} \quad H_3 = \begin{vmatrix} -1 & -2 & -1 \\ -2 & 12 & -1 \\ -1 & -2 & -1 \end{vmatrix}. \tag{3.9}$$

Der Laplace-Operator hat eine erhöhte Rauschempfindlichkeit und ist deswegen auch anfälliger gegenüber Bildstörungen. Deutlich zu erkennen ist dies auch am Beispielbild 3.5 . Vor der eigentlichen Verarbeitung mittels des Laplace-Operators sollte das Bild also gefiltert werden um störendes Bildrauschen zu minimieren.

3.1.4 Lokalisierung mittels "SUSAN"

SUSAN steht für "Smallest Univalue Segment Assimilating Nucleus" und ist ein relativ neuer Ansatz zur Detektion von Kanten (1-D feature detection) und Ecken (2-D feature detection). Desweiteren sind Filter zur strukturerhaltenden Rauschreduktion implementiert [15].

Der SUSAN Kanten-Detektor. Der Kanten-Detektor Algorithmus ist in mehrere Teilschritte gegliedert. Der SUSAN Kanten-Detektor nutzt kreisförmige Masken, auch Kernel genannt. Der Radius der Masken beträgt beispielsweise 3,4 Pixel, was einer Maskenfläche von 37 Pixeln entspricht. Diese Masken werden auf jeden Bildpunkt des Testbildes gesetzt und die Helligkeit eines jeden Pixels des Bildes innerhalb der Maske wird mit dem des Nucleus verglichen. Der Nucleus repräsentiert den zentralen Bildpunkt, also jenen Bildpunkt, auf den gerade die Maske gesetzt wird. Die einzelnen Berechnungen sind durch die Gleichungen unter 3.10 auf der Seite 23 gegeben [15]. Dieser Vergleich mit dem Nucleus wird mit allen Bildpunkten innerhalb der Maske durchgeführt:

$$n(\overrightarrow{r_0}) = \sum_{\overrightarrow{r}} c(\overrightarrow{r}, \overrightarrow{r_0}) \,. \tag{3.11}$$

n gibt den Flächenanteil innerhalb der Maske an, der die gleiche beziehungsweise eine ähnliche Helligkeit (in Abhängigkeit von t) wie der Nucleus hat. Diese Fläche wird als USAN-Fläche bezeichnet, wobei USAN für "Univalue Segment Assimilating Nucleus" steht. Abschließend wird n mit einem festen Schwellenwert g verglichen,

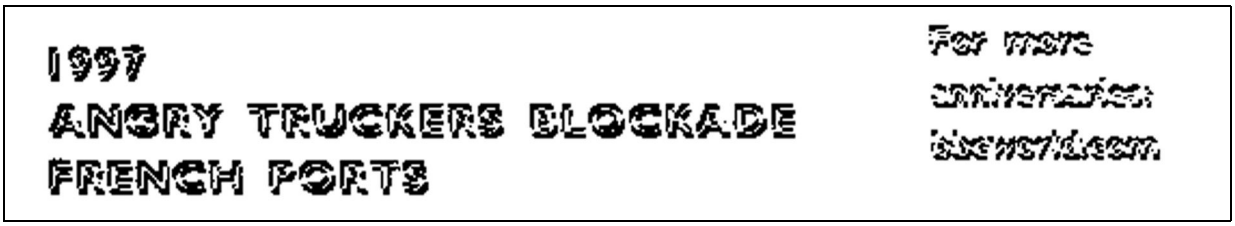


Abbildung 3.1: SUSAN, Ausschnitt von Testvorlage 004.bmp

wobei für g gilt: $g = 3n_{max}/4$ und n_{max} gleich der Anzahl Pixel innerhalb der Maske ist. Der Vergleich wird durch folgende Gleichung repräsentiert:

$$R(\overrightarrow{r_0}) = \begin{cases} \text{g - n}(\overrightarrow{r_0}) & \text{für} \quad n(\overrightarrow{r_0}) < g \\ 0 & \text{sonst.} \end{cases} \tag{3.12}$$

Bei Bildern die frei von Bildrauschen sind könnte der Schwellenwert g theoretisch entfallen.

Einen etwas glatteren Übergang zwischen dem Hintergrund und den gefundenen Kanten gibt die Gleichung 3.13.

$$c(\overrightarrow{r}, \overrightarrow{r_0}) = e^{-(\frac{I(\overrightarrow{r}) - I(\overrightarrow{r_0})}{t})^6} . \tag{3.13}$$

Bild 3.1 zeigt einen Ausschnitt der Testvorlage 004.bmp der mittels SUSAN verarbeitet wurde. Der SUSAN Kanten-Detektor wurde testweise im selbstentwickelten Programm verwendet (siehe 4.2 - SUSAN). Hierbei wurden die oben genannten Parameter verändert um möglichst nur Kanten von Textstellen zu erfassen. Für eine genauere weitergehende Betrachtung der Algorithmen sei auf [15] verwiesen.

$$c(\overrightarrow{r}, \overrightarrow{r_0}) = \begin{cases} 1 & \text{für } |I(\overrightarrow{r}) - I(\overrightarrow{r_0})| \leq t \\ 0 & \text{für } |I(\overrightarrow{r}) - I(\overrightarrow{r_0})| > t \end{cases} \tag{3.10}$$

c:	Ergebnis des Vergleichs zwischen dem Nucleus und einem anderen Bildpunkt,
$\overrightarrow{r}$:	Position des Nucleus,
$\overrightarrow{r_0}$:	Position der anderen Punktes innerhalb des Kernels,
$I(\overrightarrow{r})$:	Helligkeit an der Position r,
t:	Helligkeits-Schwellenwert, empirisch ermittelt und in[15] mit $\pm$27 Graustufen angegeben.

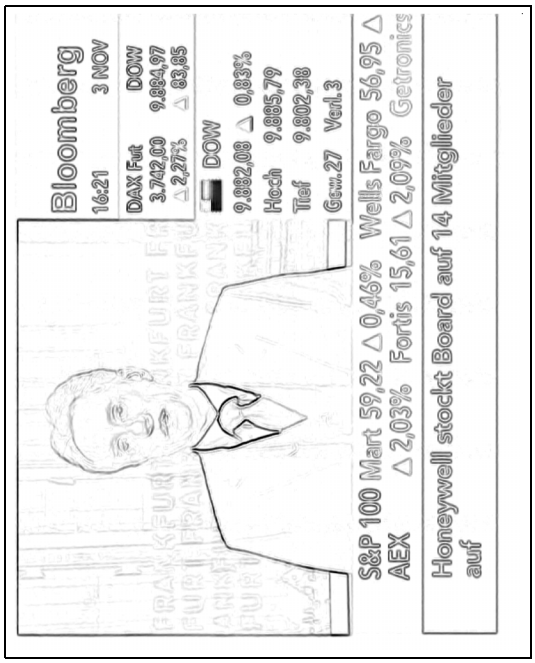


Abbildung 3.2: Beispiel Gradient-Filter

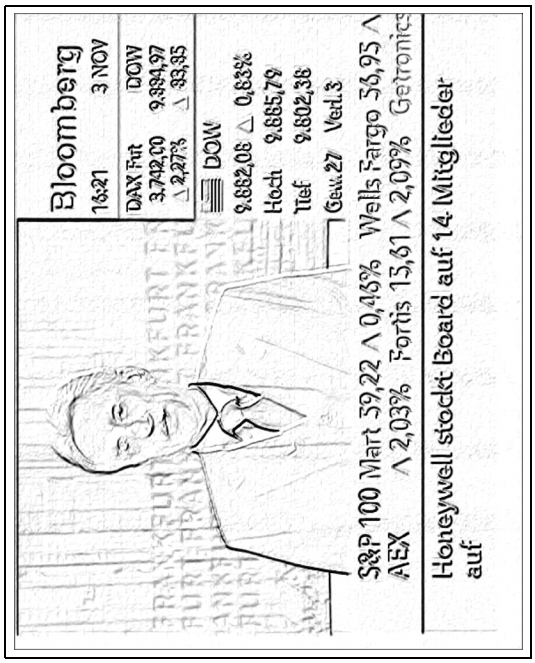


Abbildung 3.3: Beispiel Sobel-Filter

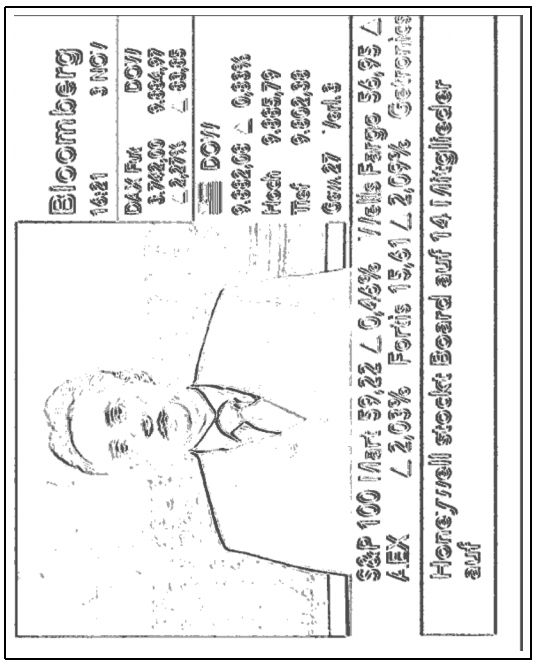


Abbildung 3.4: Beispiel Susan-Filter

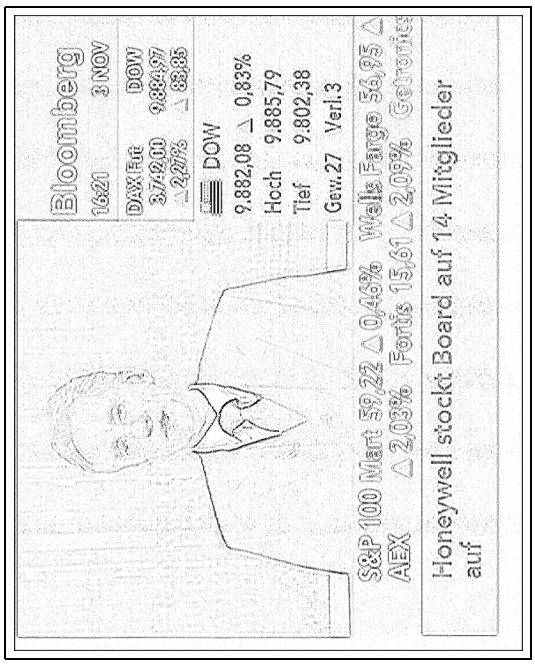


Abbildung 3.5: Beispiel Laplace-Filter

3.2 Extraktion der Textbereiche

Durch die Kantenoperatoren werden die Gebiete mit Kanten in den Bildern detektiert. Gebiete mit einer hohen Anzahl von Kanten sind mögliche Textbereiche. Da es das Ziel ist diese Textbereiche in den Videobildern zu finden, kann speziell nach Gebieten mit einer hohen Anzahl von Kanten gesucht werden. Mit einem geeigneten Schwellenwert zu dem jeweiligen Kantenoperator entstehen Binärbilder. Durch Summation der Übergänge zwischen den beiden Grauwerten in Zeilenrichtung, also durch die Addition der Anzahl der Kanten in dieser Richtung können die Textzeilen in den Videobildern gefunden werden.

Die Textzeilen sind hierbei dadurch gekennzeichnet, dass sie eine wesentlich höhere Anzahl an Kanten enthalten als die direkte Umgebung. Als Beispiel zeigt die Abbildung 3.6 im oberen Teil einen Ausschnitt der Testvorlage 004.bmp. Im mittleren Teil ist das dazugehörige Kantenbild zu erkennen wie es vom entwickelten C++ Programm ausgegeben wird. Unten und rechts sind die Projektionen des Kantenbildes dargestellt. Eine einzelne Zeile ist deutlich zu erkennen. Da Buchstaben eine wesentlich größere Höhe als 1 Pixel haben, wurden bei dem Beispielbild einzeln vorkommende Peaks bei der horizontalen Projektion entfernt. Daneben wurden bei der Projektion Gebiete unterhalb eines gewissen Schwellenwertes auf Null gesetzt. Nachdem die einzelnen Zeilen Z_n gefunden sind, kann in vertikaler Richtung auf

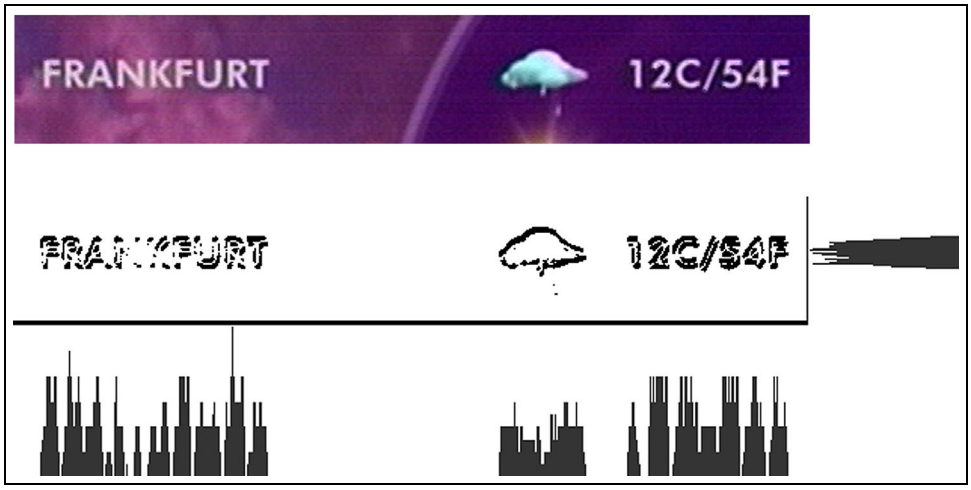


Abbildung 3.6: Projektion der Kanten, Ausschnitt vom Testbild 007.bmp

gleiche Weise vorgegangen werden. Die einzelnen Zeilen $Z_0..Z_n$ werden nacheinan-

der verarbeitet. In den einzelnen Zeilen werden die Bereiche mit Text durch die Projektion in vertikaler Richtung gesucht. Dadurch erhält man die Koordinaten der einzelnen Textbereiche innerhalb des Bildes. Betrachtet man die Abbildung 3.6, so erkennt man unterhalb des Bildes in der Projektion der Übergänge deutlich die einzelnen Bereiche der Buchstaben und Wörter.
Letztendlich hat auch die Wahl des Kantenoperators einen großen Einfluß auf das Projektionsprofil. Die Abbildungen 3.7 und 3.8 zeigen den Einfluß der unter 3.1.1 und 3.1.3 genannten Kantenfilter auf die Projektion der Kanten. Wie erwähnt besitzt der Laplace-Operator eine erhöhte Rauschempfindlichkeit. Im der unteren Abbildung ist die Auswirkung dieser Rauschempfindlichkeit bei der Verarbeitung der Testvorlage und somit auch auf die Projektion der Kanten deutlich zu erkennen.

3.3 Interpolation der Textbereiche

Mittels Interpolation ist es möglich die Auflösung der Bilder und damit die Datenmenge zu erhöhen. Eine Erhöhung der Auflösung kann zu einer Verbesserung der Erkennungsleistung der OCR-Software führen. Wie unter 2.3.2 bereits beschrieben, bieten einige Standard OCR-Programme beim Einlesen eine automatische Erhöhung der Auflösung der Vorlagen an, andere jedoch lassen eine Verarbeitung von Bildern mit zu geringer Auflösung nicht zu. Deshalb ist es sinnvoll, bereits vor der Verarbeitung der Testvorlagen mittels OCR-Software die Auflösung zu erhöhen. Insbesondere bei kleinen Schriftgrößen in den Videobildern kann dies zu einer Verbesserung der Erkennungsleistung führen. Im entwickelten Programm geschieht die Erhöhung der Auflösung vor der Binarisierung der Testvorlagen. Für die Erhöhung der Auflösung lassen sich wie im Folgenden beschrieben verschiedene Interpolations-Methoden in der Literatur finden:

Nearest Neighbor:

Dies ist ein sehr einfacher Algorithmus mit dem Vorteil sehr wenig Rechenleistung zu benötigen. Der hinzuzufügende neue Pixel bekommt den Wert des nächsten Nachbarpixels, weshalb man diese Methode auch unter dem Namen "Pixelwiederholung"

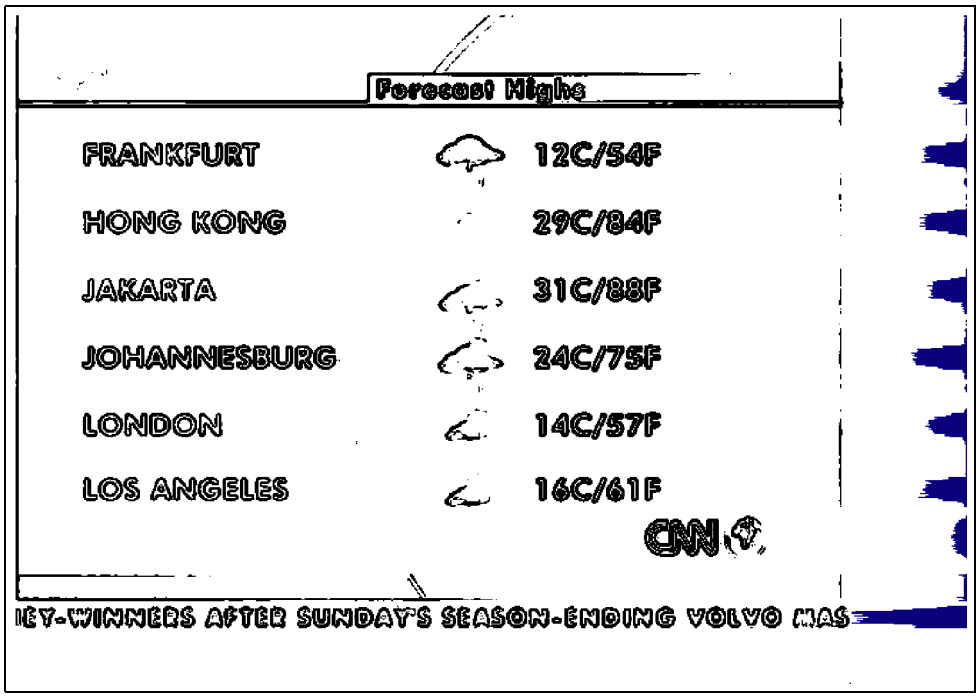


Abbildung 3.7: Beispiel Projektion der Kanten -Gradient-

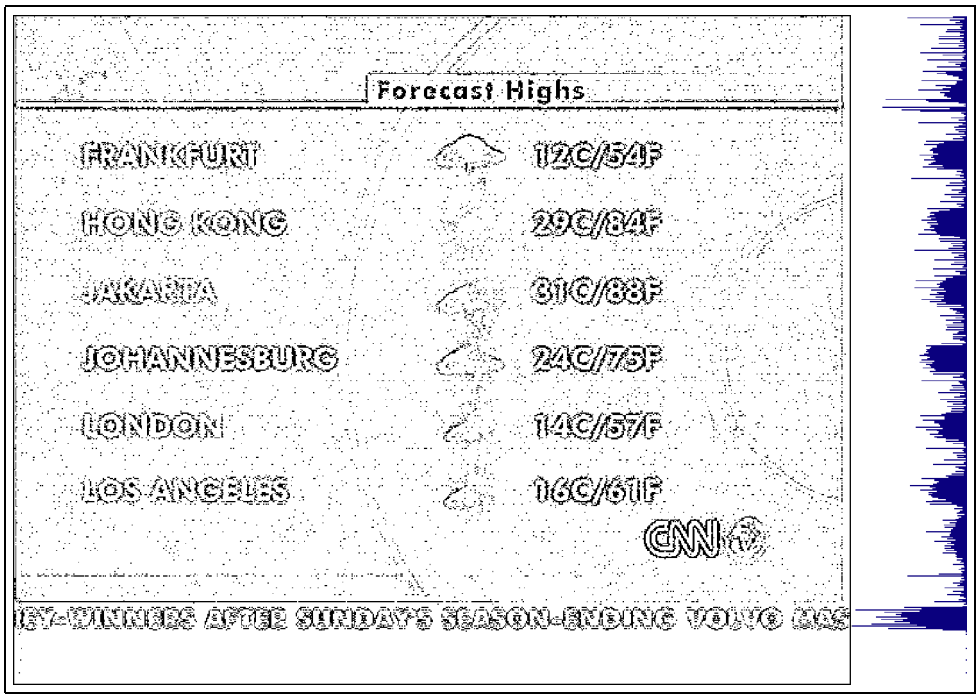


Abbildung 3.8: Beispiel Projektion der Kanten -Laplace-

in einigen Programmen wiederfindet. Ein großer Nachteil dieser Methode ist, dass die Bilder sehr schnell kantig und "ausgefranst" aussehen und insbesondere schräg verlaufende Kanten schnell stufig wirken.

Bilineare Interpolation:

Bei der bilinearen Interpolation wird der Grauwert der neuen Pixel gewichtet und aus den benachbarten Bildpunkten des Originalbildes ($B_1..B_4$) berechnet:

$$R = (1-x)(1-y)B_1 + (1-x)yB_2 + (1-y)xB_3 + xyB_4\,. \tag{3.14}$$

$B_1..B_4$:	Grauwerte der Bildpunkte des Originalbildes,
R:	Neuer Bildpunkt an der Stelle (x, y),
x, y	Position des neuen Bildpunktes zwischen den Punkten $B_1..B_4$, mit $0 \leq (x, y) \leq 1$.

Durch die Mittelung von Grauwerten können im Bild unter Umständen neue Grauwerte auftreten, die vor der Interpolation nicht vorhanden waren. Die Mittelung hat überdies auch grauwertglättende Eigenschaften. Die Grauwertübergänge werden etwas gedämpft, das Bild erscheint unschärfer.

Bikubische Interpolation:

In die bikubische Interpolation, auch kubische Faltung (cubic convolution) genannt, geht im Gegensatz zur bilinearen Interpolation bei der Berechnung der einzelnen neuen Pixel eine Umgebung von insgesamt 16 Bildpunkten ein. Bei der Wahl geeigneter Parameter kann der Tiefpaßeffekt, also die Dämpfung der Grauwertübergänge bei der bilinearen Interpolation, vermieden werden.

Abbildung 3.9 auf Seite 29 zeigt von links nach rechts: Nearest Neighbor, Bilineare Interpolation, Bikubische Interpolation. Deutlich zu erkennen sind die stufigen Kanten bei der nearest Neighbor-Interpolation.

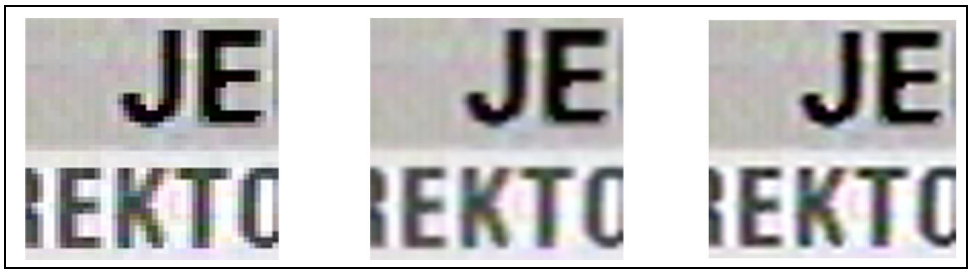


Abbildung 3.9: Beispiel Nearest Neighbor, Bilineare Interpolation, Bikubische Interpolation

3.4 Binarisierung der Textbereiche

Für eine optimale Vorbereitung der gefundenen Textstellen auf die Verarbeitung in kommerzieller OCR-Software sollte im Bild die Schrift in schwarz vor weißem Hintergrund erscheinen. Dies kann durch die Binarisierung der gefundenen Textbereiche erfolgen. Nach der Umwandlung der Farbvorlagen in Graustufenbilder erstellt man ein Binärbild das nur zwei Grauwerte enthält, nämlich Schwarz = 0 und Weiß = 255. Die einfachste Lösung hierfür stellt eine konstante Grauwertschwelle G_S dar mit $0 \leq G_S \leq 255$ und der Grauwerttransformation:

$$y = \begin{cases} 0 & \text{für } x < G_S \\ 255 & \text{sonst.} \end{cases} \tag{3.15}$$

Um die Grauwertschwelle G_S zu ermitteln bieten sich verschiedene Methoden an. Dabei ist es wichtig einen möglichst genauen Schwellenwert zu finden. In der Abbildung 3.10 können die Auswirkungen unterschiedlicher Schwellenwerte erkannt werden. Im linken Bild ist der Schwellenwert zu niedrig angesetzt, Teile des Hintergrundes werden auf Schwarz gesetzt. Im rechten Bild ist der Schwellenwert zu hoch angesetzt, Teile der Schrift werden als Hintergrund interpretiert und gehen verloren. In beiden Fällen, insbesondere wenn Teile des Hintergrundes schwarz erscheinen, kommt es zu Fehlern bei der Texterkennung mit der OCR-Software. Im mittleren Bild wurde ein optimaler Schwellenwert gefunden; der Text und der Hintergrund sind klar voneinander getrennt.

Eine Möglichkeit zur Schwellenwertbestimmung ist die Auswertung des Histogramms. In der Abbildung 3.11 sind zwei unterschiedliche Bereiche des Bildes markiert; ein

Abbildung 3.10: Beispiel Auswirkung verschiedener Grauwertschwellen, Ausschnitt vom Testbild 004.bmp

Bereich oberhalb und ein Bereich innerhalb des gefundenen Textbereiches. Daneben sind jeweils die Histogramme der markierten Bereiche dargestellt. Im Bildbereich außerhalb der Schrift dominieren deutlich einige wenige Grauwerte des Hintergrundes die Verteilung des Histogramms. Im Textbereich des Bildes ist dieses Peak im Histogramm auch zu erkennen, zusätzlich sind die dunkleren Grauwerte des Textes im Histogrammverlauf zu sehen. Bei Versuchen mit unterschiedlichen Textbereichen in verschiedenen Testvorlagen hat sich ein Schwellenwert als gut geeignet erwiesen der etwas kleiner, beziehungsweise bei heller Schrift vor dunklem Hintergrund etwas größer als der geometrischen Mittelwert der Grauwerte des Textbereiches ist. Dies führt zum Teil dazu, dass einige Pixel im Übergangsbereich zwischen Hintergrund und Buchstaben nicht den Buchstaben zugeordnet werden, sondern als Teil des Hintergrundes interpretiert werden. Dadurch wirken die Buchstaben teilweise ein wenig schlanker, behalten aber ihre Form und weisen bei der Verarbeitung in der OCR-Software dadurch teilweise weniger Fehler auf. Grundsätzlich ist bei dieser Vorgehensweise der Schwellenwertfindung davon auszugehen, dass der Hintergrund im Bild aus ähnlichfarbigen Pixeln besteht. Bei der Umwandlung in ein Graustufenbild ergibt sich dann eine Fläche mit ähnlichen Graustufen.
Problematischer ist das Auffinden eines Schwellenwertes für Textbereiche in denen der Hintergrund nicht homogen ist, also sehr unterschiedliche Farben und/oder Kontraste aufweist. Bei der Binarisierung dieser Textbereiche kann es relativ schnell zu Fehlern kommen.

In der Literatur sind je nach Anwendung außerdem einige andere Methoden zu Ermittlung eines optimalen Schwellenwertes zu finden. In [12] wird die Schwellenwertbestimmung mit Hilfe eines modifizierten Histogramms vorgestellt, wobei das modifizierte Histogramm hierbei nur Bildpunkte aus dem Inneren des Objektes und aus dem Hintergrund enthalten soll. Bildpunkte im Übergangsbereich zwischen dem

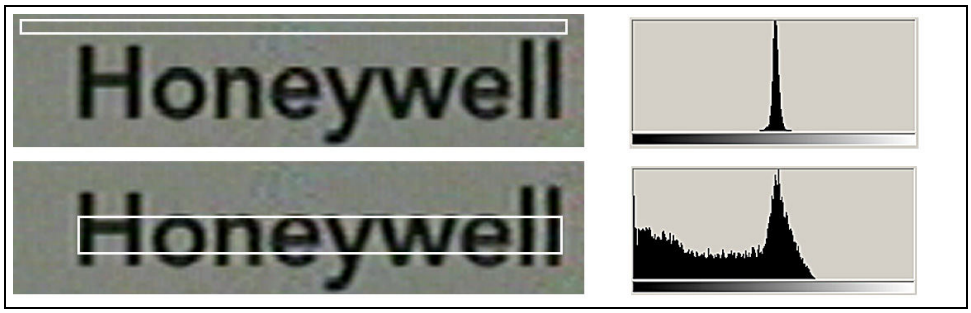


Abbildung 3.11: Beispiel Grauwertschwelle / Histogramm, Ausschnitt vom Testbild 012.bmp

Objekt und dem Hintergrund, also die Kantenbereiche, werden nicht berücksichtigt. Hierzu wird das Histogramm nur aus Bildpunkten mit einem niedrigen Wert des Gradienten gebildet. Es enthält im optimalen Fall zwei deutliche Peaks; ein Maximum für die Grauwerte des Hintergrundes und ein Maximum für die Grauwerte der Buchstaben. Der Schwellenwert für die Trennung zwischen Hintergrund und Textbereich kann dann im mittleren Bereich zwischen diesen beiden Maxima vermutet werden.

Bei Versuchen mit den Texten in den Testvorlagen hat sich eine etwas andere Vorgehensweise zur Ermittelung des Schwellenwertes als sehr gut geeignet erwiesen. So ist in der Abbildung 3.12 ein Buchstabe aus einer Testvorlage stark vergrößert dargestellt. Im mittleren Teil der Abbildung sind die Pixel im Übergangsbereich zwischen Hintergrund und Buchstabe bläulich markiert. Diese Pixel enstehen zum Teil bei der Interpolation der Textbereiche (siehe 3.3) und haben einen Grauwert, dessen Durchschnitt zwischen dem des Hintergrundes und dem Grauwert im unmarkierten Bereich des Buchstaben liegt. Der Schwellenwert G_s läßt sich also nach der Formel:

$$G_s = \frac{\sum \text{Grauwerte im blauen Bereich}}{\text{Anzahl der Pixel im blauen Bereich}} \tag{3.16}$$

berechnen. Der bläulich markierte Bereich in Abbildung 3.12 entspricht dem Bereich der Kante zwischen Hintergrund und Buchstabe und läßt sich mit Hilfe der unter 3.1.1 erwähnten Kantenfilter bestimmen. Im rechten Teil der Abbildung 3.12 ist der einzelne Buchstabe nach der Binarisierung dargestellt.

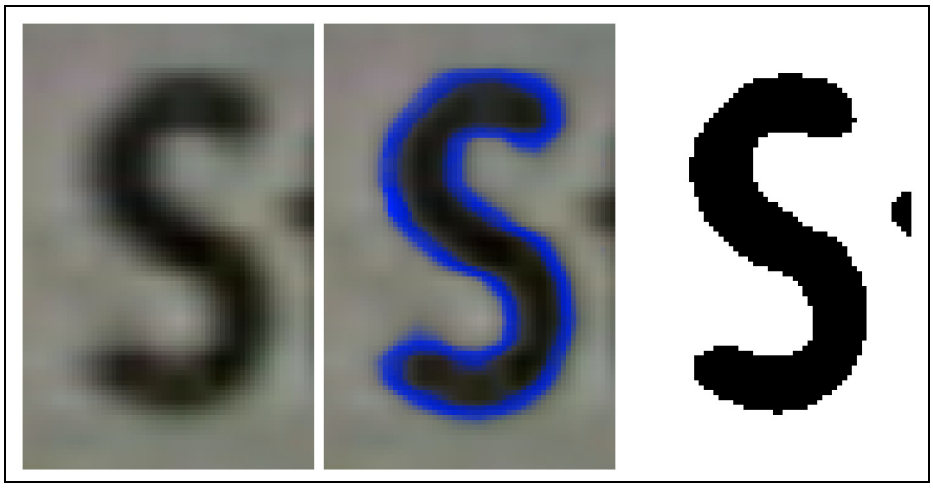


Abbildung 3.12: Beispiel Grauwertschwelle im Kantenbereich

Unterscheidung zwischen Positiv- und Negativ-Schrift:

Bisher wurde bei der Ermittlung des Schwellenwertes nicht berücksichtigt, dass die Unterteilung der Schrift und des Hintergrundes in Positiv- und Negativ-Vorlagen erfolgen kann: die Schrift kann im Original hell vor dunklem Hintergrund oder dunkel vor hellem Hintergrund erscheinen. Wenn es sich um helle Schrift vor dunklem Hintergrund handelt muss das Bild invertiert werden damit nach der Binarisierung wieder schwarze Schrift vor weißem Hintergrund vorhanden ist. Ein einfaches Kriterium hierfür ist der Vergleich der beiden Mittelwerte der Grauwertbereiche innerhalb und außerhalb des Textbereiches. Falls der Mittelwert außerhalb des Textbereiches größer ist als der innerhalb des Textbereiches muss das Bild invertiert werden, da die Umgebung um den Text in diesem Fall dunkler ist als der Text selber. Ein Beispiel hierfür zeigt die Abbildung 3.13; im Gegensatz zum unteren Teil ist im oberen Teil der Abbildung die Schrift heller als der Hintergrund und muss deshalb invertiert werden.

3.5 Reduktion von Fehlern

Bei der Verarbeitung der Testvorlagen oder auch anderer Videobilder treten unterschiedliche Fehler bei der Verarbeitung auf. Unter anderem sind folgende Fehler zu beobachten:

- Bildbereiche mit Text werden nicht erkannt und in der später erfolgenden Texterkennung nicht weiterverarbeitet.
- Bildinhalte ohne Text werden als Textbereiche erkannt.
- Erkannte Textbereiche sind zu klein, so dass einzelne Buchstaben oder Ziffern abgeschnitten und fehlerhaft verarbeitet werden.
- Es treten Fehler bei der Verarbeitung durch die kommerzielle OCR-Software auf.

Bei der unter 3.2 beschriebenen Projektion der Kanten und der damit verbundenen Lokalisierung von Textstellen in den Testvorlagen kann es vorkommen, dass einzelne Buchstaben und nicht nur ganze Wörter lokalisiert werden. Abbildung 3.14 zeigt dazu ein Beispiel. Im oberen Bereich sind einzelne Lücken zwischen Buchstaben zu erkennen. Normalerweise führen diese Lücken zwischen den einzelnen Buchstaben nicht zu Fehlern bei der Texterkennung, da jeder Buchstabe für einen einzelnen Textbereich gehalten und problemlos weiterverarbeitet wird. Wenn diese Lücken in einem weiteren Verarbeitungsschritt wieder entfernt werden, besteht später die Möglichkeit Textbereiche zu verarbeiten die komplette Wörter enthalten. Diese können dann beispielsweise als einzelne Bilddateien abgespeichert werden oder auch direkt einer OCR-Software für die Texterkennung übergeben werden. In der Abbildung 3.14 zeigt der untere Teil den Textbereich nach Entfernung der vertikalen Lücken. Da nach Möglichkeit größere Lücken (Leerzeichen) zwischen einzelnen Wörtern erhalten bleiben sollen, dürfen nur vertikale Lücken bis zu einer maximalen Breite B_{max} entfernt werden, wobei B_{max} von der Schriftgröße und der Auflösung der Videobilder abhängt. In den Testvorlagen hat sich $B_{max} = 6$, also ein maximaler Wert für die

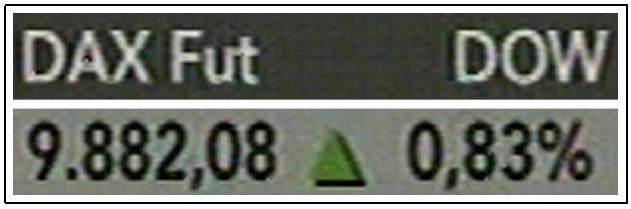


Abbildung 3.13: Beispiel Positiv- und Negativ-Schrift, Ausschnitt vom Testbild 012.bmp

vertikale Lücke von 6 Pixeln als gut geeignet erwiesen. Sind die Lücken zwischen

Abbildung 3.14: Beispiel Textlücken, Ausschnitt vom Testbild 007.bmp

einzelnen Buchstaben geschlossen, können in einem weiteren Verarbeitungsschritt Textbereiche entfernt werden, die in ihren Proportionen von normalen Verhältnissen bei Schriften abweichen. Von einer Weiterverarbeitung können unter Berücksichtigung der unter 2.1 genannten Punkte jene Textbereiche ausgeschlossen werden für die gilt:

- $Hoehe > Breite$. Wörter oder Ziffernfolgen weisen immer einer größere Breite als Höhe auf.
- $Texthoehe \leq 8Pixel$. Werden die Schriften in den Testvorlagen mit einer Auflösung von $720 * 576$ Pixeln betrachtet, kann festgestellt werden, dass die kleinsten enthaltenen Schriften im Bereich von 10 bis 16 Pixel liegen. Für andere Auflösungen müßte dieser Wert entsprechend angepaßt werden.
- $Texthoehe \geq 0,5 * Bildhoehe$. Texte die größer sind als die halbe Bildhöhe können ebenfalls entfernt werden.

3.6 Redundanz-Ausnutzung im Videodatenstrom

Die verwendeten Testbilder sind Einzelbilder, die wie unter 2.2 beschrieben erstellt wurden. Beim PAL Standard werden Videobilder mit 50 Halbbildern je Sekunde übertragen, das entspricht 25 Vollbildern je Sekunde. Damit ein Text für den Betrachter gut lesbar ist, sollte er für mindestens 0.75 Sekunden im Bild zu sehen sein [11], so dass ein Text in mindestens 18 aufeinanderfolgenden Videobildern zu finden sein muss. Folglich muss nicht jedes einzelne Videobild auf das Vorhandensein von Text untersucht werden. Es reicht aus in den Videobildern mit einem gewissen

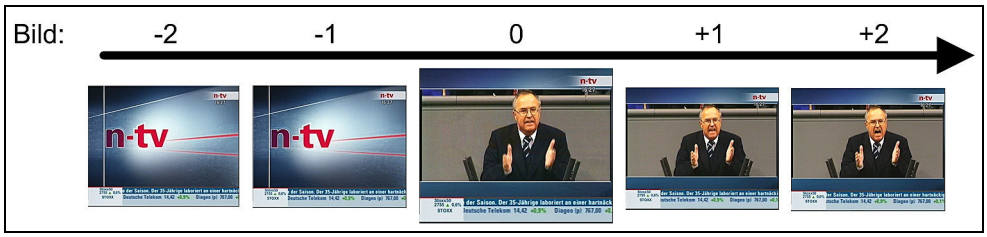


Abbildung 3.15: Zeitlicher Ablauf der Videobilder

zeitlichen Abstand nach vorhandenen Texten zu suchen. Sind in einem einzelnen Videobild alle Textbereiche mit den dazugehörigen Koordinaten im Bild gefunden (siehe 3.2), so können Videobilder mit einem zeitlichen Abstand $\pm t$ zu dem Einzelbild dazu dienen, mögliche Fehler die bei der Detektion der Textbereiche auftreten können zu reduzieren. Folgende Vorteile gegenüber der Verarbeitung einzelner Bilder ergeben sich:

- Durch die Verarbeitung mehrerer Bilder kann die Erkennungsrate für die Textbereiche gesteigert werden. Ein Textbereich, der in einem ersten Verarbeitungsschritt im Bild n unerkannt blieb, kann durch Verarbeitung von Bildern im zeitlichen Abstand $\pm t$ unter Umständen dennoch gefunden werden.

- Textbereiche die nur in einem Bild gefunden werden, aber in Bildern mit dem zeitlichen Abstand $\pm t$ nicht enthalten sind, können verworfen werden. Hierbei handelt es sich sehr wahrscheinlich nicht um Text, sondern um eine Fehlinterpretation anderer Bildinhalte.

- Die Textbereiche im Bild können genauer ermittelt werden. Die Eckpunkte für die einzelnen Textbereiche $(X_{ol}, Y_{ol}), (X_{ur}, Y_{ur})$ (oben links, unten rechts), können durch die zeitliche Mittelung über mehrere Bilder genauer festgelegt werden.

- Die Rechenzeit für die Bearbeitung kann verringert werden; anstatt jedes einzelne Videobild zu verarbeiten kann ein Teil der Bilder ausgelassen werden, ohne dass deswegen Text nicht erfaßt wird.

Im zeitlichen Verlauf über mehrere Bilder ist bei der Auswertung der Textbereiche zu beachten, dass sich die Position des Textbereiches im Bild verschiebt. Abbil-

Abbildung 3.16: Beispiel Bewegung von Text

dung 3.16 zeigt hierzu ein Beispiel. In der Testvorlage sind zwei Bereiche hell umrandet markiert. Während der obere Schriftzug "Bloomberg" unverändert immer an einer Position stehen bleibt, weist der untere Schriftzug "Electric" in den nachfolgenden Bildern einen horizontalen Versatz nach rechts auf. Wie unter 2.1 bereits erwähnt, sind neben einer horizontalen Bewegung natürlich auch vertikale Bewegungen der Texte im Bild möglich. Die Größe, also die Anzahl der Pixel aus der ein Textbereich besteht, bleibt unabhängig von einer möglichen Bewegung immer konstant.

Auf der CD im Anhang befindet sich eine zweite Version des selbst entwickelten C++ Programms. Diese Version demonstriert beispielhaft die mögliche Nutzung mehrerer Fernsehbilder in Folge anhand einer kurzen Bildsequenz aus dem Programm des Fernsehsenders N-TV. Es sind keine weiteren Eingaben nach Aufruf des Programms nötig. Das Programm vergleicht die Anzahl der Kanten des mittleren Bildes als Bezugsbild, mit je drei Fernsehbildern davor und danach. Bei der Verarbeitung und Ausgabe des mittleren Bildes werden letztendlich lediglich Textbereiche berücksichtigt die in allen sieben Bildern, also in einem zeitlichen Bereich von circa 0,28 Sekunden zu finden sind.

Kapitel 4

Aufbau des C++ Programms

Das entwickelte Programm ist mit Hilfe des Microsoft Visual C++ Compilers 6.0 unter Windows XP erstellt worden. Im Anhang ist der Quellcode des Programms zu finden. Der gesamte Quellcode ist im Rahmen dieser Diplomarbeit entstanden. Der Quellcode besteht aus zwei Teilen: Der Hauptdatei und einer Includedatei (siehe Anhang). Zusätzlich ist die freie C-Bibliothek "ICE - Image C Extension" der Universität Jena unter den Bedingungen der LGPL (GNU Library General Public Licence) verwendet worden. Insbesondere grundlegende Operationen wie das Öffnen, Lesen und Schreiben von Bilddateien oder auch der Zugriff auf einzelne Bildpunkte der eingelesenen Daten wurden hiermit ausgeführt. Für eine genauere Beschreibung der Bibliothek sei auf [20] verwiesen. Getestet und lauffähig ist die Executable Version des Programms unter Windows 2000 und Windows 98.

4.1 Flow Chart

Abbildung 4.1 zeigt das Flow Chart zum Programm. In dem Chart ist der Ablauf der Verarbeitung der Testvorlagen dargestellt. Für eine genaue Betrachtung der einzelnen Funktionen des Programms sei auf 4.2 und den Anhang mit dem Quellcode, insbesondere den Hauptteil des Programms (*main*) verwiesen.

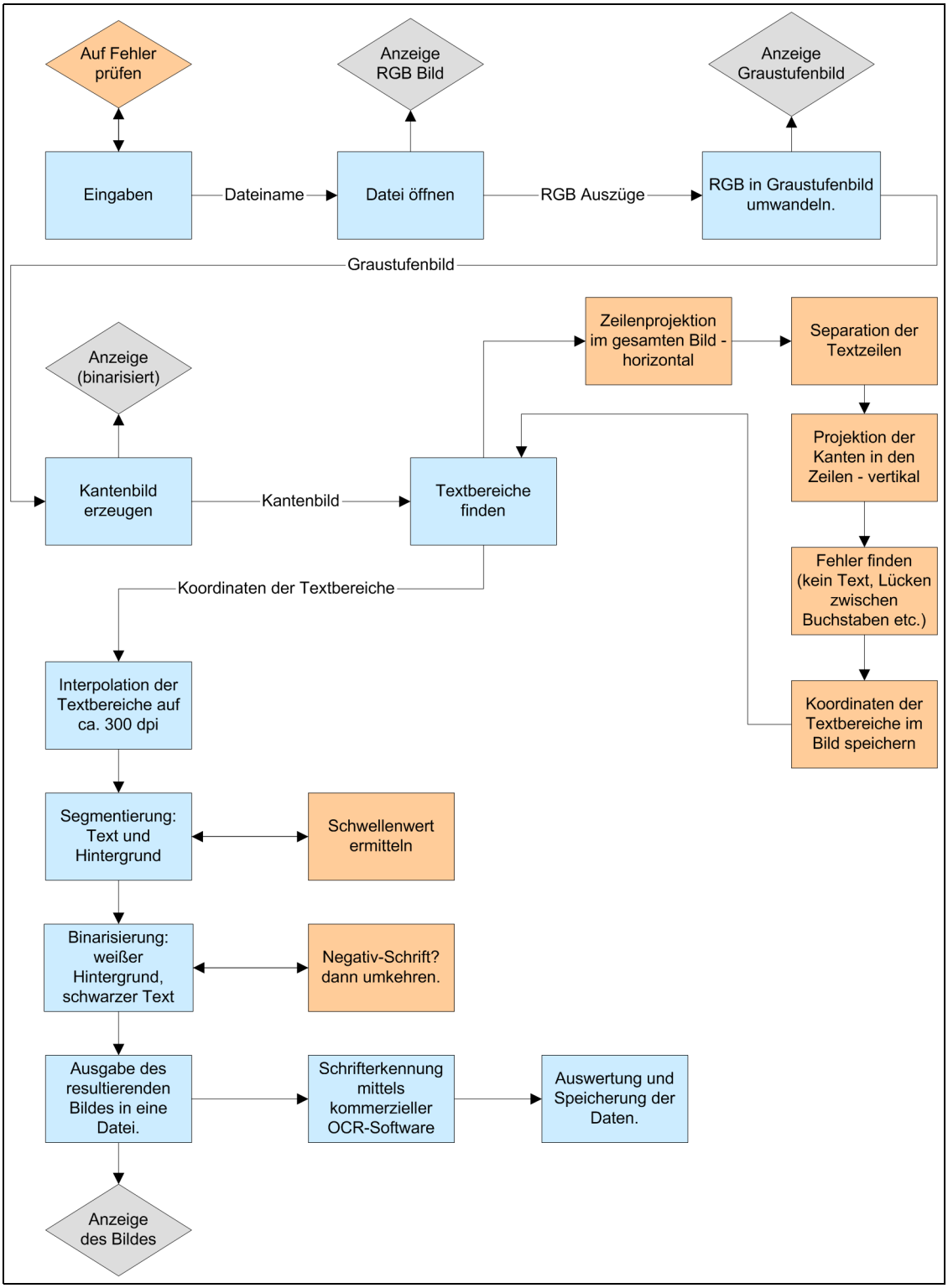


Abbildung 4.1: Flow Chart des entwickelten Programms

4.2 Die einzelnen Funktionen im Detail

Hier sei eine kurze Übersicht und Beschreibung der wichtigsten Funktionen des Programms gegeben:

function RGBOpen: Öffnet eine Bilddatei und liest die RGB-Daten in 3 Variablen des Types *Image* ein. Mögliche Formate für die einzulesende Datei sind: .jpg, .jpeg, .bmp und .tga. Die Auflösung der Bilder ist hierbei beliebig.

function RGBtoGrau: Gibt eine Variable des Types *Image* zurück. Übergeben und in ein Graustufenbild umgewandelt werden die drei Farbauszüge eines RGB-Bildes. Für die Umrechnung der RGB-Werte in Graustufen siehe 2.2.

function RGB-Clean: Erstellt ein RGB-Bild aus der Testvorlage, das nur die gefundenen Textbereiche enthält. Dient nur der Überprüfung während der Verarbeitung.

function ShowC: Zeigt das an die Funktion übergebene Bild an und wartet auf einen Tastendruck.

function SUSAN: Ermittelt die Kanten im übergebenen Bild. Der SUSAN Algorithmus ist unter 3.1.4 beschrieben.

function GradientBild: Ermittelt die Kanten im übergebenen Bild. Der Gradienten-Algorithmus ist unter 3.1.1 beschrieben.

function SobelFilter: Ermittelt die Kanten im übergebenen Bild. Der Sobel-Algorithmus ist unter 3.1.2 beschrieben.

function LaplaceFilter: Ermittelt die Kanten im übergebenen Bild. Der Laplace-Algorithmus ist unter 3.1.3 beschrieben.

function BilInterpol: Interpoliert den übergeben Bildbereich (siehe 3.3) um den Faktor vier.

function InterpolGrauwert: Gehört zur Funktion *BilInterpol* und berechnet die bei der Interpolation benötigten neuen Grauwerte.

function ZeilenProjection: Erstellt ein Bild der Testvorlage mit einem seitlichen Rand, der die Projektion der Anzahl der Kanten im Bild enthält. Dient nur zur Überprüfung während der Verarbeitung.

function SpaltenProjection: Gleich der Funktion *ZeilenProjection*, die Berechnung allerdings erfolgt in Richtung der Spalten.

function CleanLuecken: Entfernt vertikale Lücken zwischen einzelnen Textbereichen (Buchstaben) (vergleiche 3.5).

function CleanBuchstaben: Entfernt Textbereiche deren vertikale Ausdehnung größer als die horizontale Ausdehnung ist (vergleiche 3.5).

function CleanPixelHoehe: Entfernt Textbereiche unterhalb einer Mindesthöhe (vergleiche 3.5).

function CleanBildHoehe: Entfernt Textbereiche die eine maximale Höhe ($> 1/2$ *Bildhoehe*) überschreiten (vergleiche 3.5).

function AnzahlKantenZeilen: Legt eine Matrix der Größe (Image-YSize, 1) an und trägt die Anzahl der Kanten in horizontaler Richtung in dieser ein (vergleiche 3.2).

function AnzahlKantenSpalten: Legt eine Matrix der Größe (1, Image-XSize) an und trägt die Anzahl der Kanten in vertikaler Richtung in dieser ein (vergleiche 3.2).

function Zeilen_Finden: Sucht nach Textzeilen in der von der Funktion *AnzahlKantenZeilen* angelegten Matrix in horizontaler Richtung.

function WoerterFinden: Sucht nach den einzelnen Wörtern in der von der Funktion *ZeilenFinden* vorverarbeiteten Matrix.

function Biniarisierung: Erstellt aus den Textbereichen Binärbilder (vergleiche 3.4).

Kapitel 5

Ergebnisse und Auswertung

Nach der Ausgabe der verarbeiteten Testvorlagen durch das entwickelte Programm kann die Texterkennung mittels kommerzieller OCR-Software erfolgen. Alternativ wäre auch eine direkte OCR der einzelnen Textbereiche innerhalb des eigenen Programms denkbar. Hierfür existieren diverse kommerziell verfügbare Softwarebibliotheken mit deren Hilfe sich die Erkennung der einzelnen Wörter realisieren ließe. Da die auf dem Markt vorhandene Software zumindest im Bereich der Erkennung von Maschinenschrift bereits sehr ausgereift ist, wäre eine eigene Entwicklung von Routinen für die direkte Erkennung der Texte sicherlich nicht sinnvoll und unter Berücksichtigung der Komplexität eines solchen Vorhabens schon aus zeitlichen Gründen schwer realisierbar.

Für die Auswertung der vorhandenen Testvorlagen wurden alle Vorlagen vom selbstentwickelten Programm verarbeitet und als BMP-File abgespeichert. Die Auswertung kann in zwei Teile aufgeteilt werden: Zum Einem läßt sich ein Auswertungsergebnis für das entwickelte Programm angeben. Hierbei ist von Interesse inwieweit vorhandene Textstellen in den Testvorlagen korrekt identifiziert werden, beziehungsweise wieviele Pixel der Testvorlagen fälschlicherweise als Textpixel erkannt werden. Zum Anderen läßt sich eine Erkennungsrate der kommerziellen OCR-Software in den verarbeiteten Bildern angeben. Dieser Wert gibt Aufschluß darüber wieviele Textbereiche in den Testvorlagen insgesamt korrekt identifiziert und letztlich als normale Textdateien ausgegeben werden.

5.1 Ergebnis für die Lokalisierung der Textstellen

Als Erstes erfolgt die Verarbeitung der unterschiedlichen Testvorlagen mittels des entwickelten C++ Programms. Man erhält hierdurch entsprechend der Anzahl der Testvorlagen 30 weitere Bilddateien. Aus diesen Bilddaten läßt sich sehr schnell mit Hilfe des Histogramms die Anzahl der Pixel bestimmen, die vom Programm als Textbereiche interpretiert wurden; dies sind alle schwarzen Pixel. Um zu ermitteln wieviele Pixel der Testvorlagen vom entwickelten Programm fälschlicherweise den Textbereichen zugeordnet wurden, können mit Hilfe eines Bildbearbeitungsprogramms alle Pixel entfernt werden, die nicht zu Buchstaben, Ziffern und dergleichen gehören. Durch Differenzbildung aus der Anzahl der schwarzen Pixel vor und nach dem Entfernen dieser Bereiche läßt sich sowohl die Anzahl der Pixel ermitteln, die korrekt identifiziert wurden, als auch die Anzahl jener Pixel, die fälschlicherweise als als Text interpretiert wurden.
Für den Vergleich mit den Originaltestvorlagen wurden auch diese mit Hilfe eines Bildbearbeitungsprogramms modifiziert. Äquivalent zu den Verarbeitungsschritten des entwickelten Programms wurden die Originalbilddaten zuerst in ein Graustufenbild konvertiert und um den Faktor vier interpoliert. Manuell wurde dann für die einzelnen Texte ein Schwellenwert gesetzt und der Hintergrund aus den Bilddaten entfernt. Hieraus läßt sich wiederum mit Hilfe des Histogramms die Anzahl der gesamten vorhandenen Textpixel in den Testvorlagen ermitteln.
Die Anzahl der gesamten Pixel in den Testvorlagen beträgt nach der Interpolation durch das entwickelte Programm 6.635.520 Pixel ($720 * 576$ Pixel der Originalvorlagen vierfach interpoliert). Die Tabelle 5.1 zeigt die Ergebnisse für die einzelnen Testvorlagen Nummer 001 bis 030. Zum Teil finden sich in der Tabelle für Q_l, also die prozentuale Anzahl der korrekt aufgefundenen Textpixel durch das Programm, Werte knapp über 100%. Dies erklärt sich daraus, dass die Testvorlagen sowohl wie oben beschrieben einmal manuell verarbeitet, als auch einmal mit Hilfe des entwickelten Programms verarbeitet wurden. Bei der manuellen Verarbeitung kann es vorkommen, dass der Schwellenwert zum Extrahieren der Buchstaben aus dem Hintergrund anders gesetzt wird als das entwickelte Programm ihn setzt. Dadurch werden die Buchstaben unter Umständen ein wenig schmaler als die vom Programm extrahier-

$$Q_l = \frac{100}{P_v} * P_l \quad [\%] \tag{5.1}$$

$$Q_f = \frac{100}{P_g} * P_f \quad [\%] \tag{5.2}$$

P_v: Anzahl der tatsächlich vorhandenen Textpixel in den Testvorlagen,

P_l: Anzahl der vom entwickelten Programm korrekt lokalisierten Textpixel,

P_f: Anzahl der Bildpixel die fälschlicherweise vom Programm als Text identifiziert wurden,

P_g: Gesamtanzahl aller Pixel in den Testvorlagen nach der Verarbeitung durch das entwickelte C++ Programm (= 6.635.520),

Q_l: Quote in Prozent der korrekt lokalisierten Textpixel,

Q_f: Quote in Prozent für als Textpixel fehlinterpretierte Bildbereiche bezogen auf das Gesamtbild.

ten Buchstaben. Dennoch bleiben in beiden Fällen die Buchstaben gut lesbar, und unterscheiden sich lediglich in der Anzahl der Pixel aus denen sie bestehen (vergleiche Kapitel 3.4). Für eine Auswertung ist diese Methode trotz des subjektiven Einflusses bei der manuellen Verarbeitung der Testvorlagen sicherlich ausreichend, da der hieraus resultierende Fehler akzeptabel ist.

Nr. Testvorlage	P_v [Pixel]	P_l [Pixel]	P_f [Pixel]	Q_l [%]	Q_f [%]
001	243318	180398	136270	74,1	2,1
002	464944	368328	178494	79,2	2,7
003	37775	48180	30433	127,5	0,5
004	303302	255201	34390	84,1	0,5
005	225633	222121	0	98,4	0
006	105529	115131	208759	109,1	3,1
007	292391	289503	239372	99	3,6
008	963770	911937	0	94,6	0
009	576240	393433	278862	68,3	4,2
010	193593	114122	22829	58,9	0,3
011	126696	78458	675777	61,9	10,2
012	419493	364573	246348	86,9	3,7
013	99728	68084	0	68,3	0
014	192571	118067	12677	61,3	0,2
015	217300	178601	535086	82,2	8,1
016	103973	105745	0	101,7	0
017	102279	121825	0	119,1	0
018	207320	208595	15308	100,6	0,2
019	181706	169932	39854	93,5	0,6
020	189579	129780	104534	68,5	1,6
021	21003	19984	0	95,1	0
022	209365	205053	351810	97,9	5,3
023	141633	46054	1169	32,5	0
024	192893	211912	35418	109,9	0,5
025	344576	104250	642321	30,3	9,7
026	377380	401076	0	106,3	0
027	73257	69700	0	95,1	0
028	95721	76026	492686	79,4	7,4
029	197984	218751	625	110,5	0
030	217666	230613	116986	105,9	1,8
Mittelwert:				$\bar{Q}_l$: **84,4**	$\bar{Q}_f$: **2,2**

Tabelle 5.1: Ergebnisse der Lokalisierung der Textstellen

5.2 Ergebnis für die OCR der Textstellen

Die OCR der verarbeiteten Testvorlagen geschah mittels des Programms ABBYY Fine Reader Version 6.0. Bei der Auswertung wurden die in den Testvorlagen vorhandenen Wörter und Zeichenkombinationen manuell gezählt ($= W_g$). Nicht berücksichtigt wurden einzelne Zeichenkombinationen ohne Bezug zu anderen Wörtern oder Ziffern im Bild und angeschnittene Wörter oder Ziffern an den Bildrändern, da es sicherlich keinen Sinn macht diese Daten für eine spätere Verwendung zu erfassen. Auch die durch die OCR-Software erkannten Wörter und Zeichenkombinationen wurden manuell gezählt ($= W_e$). Wörter die entweder nicht erfaßt wurden oder mit Hilfe von Wörterbüchern nicht automatisch sinnvoll zu einem ganzen Wort ergänzt werden konnten, wurden nicht hinzugerechnet.

$$E_k = \frac{100}{\sum W_g} * W_e \quad [\%] \tag{5.3}$$

E_k: Erkennungsrate nach der Verarbeitung mittels OCR-Software in Prozent,

W_g: Anzahl der Wörter und Zeichenkombinationen in der Testvorlage insgesamt,

W_e: Anzahl der erkannten Wörter und Zeichenkombinationen nach der Verarbeitung.

Nr. Testvorlage	W_g [Wörter]	W_e [Wörter]	E_k [%]
001	25	20	80,0
002	42	37	88,1
003	2	2	100,0
004	47	43	91,5
005	4	4	100,0
006	5	4	80,0
007	27	27	100,0
008	27	23	85,2
009	37	21	56,8
010	23	15	65,2
011	13	8	61,5
012	40	34	85,0
013	7	6	85,7
014	26	15	57,7
015	26	16	61,5
016	9	9	100,0
017	6	6	100,0
018	8	7	87,5
019	27	20	74,1
020	24	14	58,3
021	2	2	100,0
022	24	13	54,2
023	16	4	25,0
024	7	7	100,0
025	11	9	81,8
026	3	3	100,0
027	3	3	100,0
028	15	12	80,0
029	14	12	85,7
030	5	5	100,0
Mittelwert:			$\bar{E}_k$: **81,1**

Tabelle 5.2: Ergebnisse der OCR der Textstellen

5.3 Auswertung der Ergebnisse

Als Gesamtergebnis erhält man einen Mittelwert für die Erkennungsrate der Textpixel durch das entwickelte Programm von $\bar{Q}_l = 84,4\%$, das heißt, dass circa 84% der in den Testvorlagen enthaltenen Textpixel durch das entwickelte Programm korrekt aufgefunden, in ein Binärbild umgewandelt und in einer neuen Datei abgespeichert werden. Bei der Auswertung der darauffolgenden OCR erhält man einen Mittelwert von 81,1% für $\bar{E}_k$. Im Mittel werden also insgesamt circa 81% der in den Testvorlagen enthaltenen Texte erkannt und extrahiert. In der Tabelle 5.1 ist besonders die fünfte und in der Tabelle 5.2 die vierte Spalte von Interesse, also die Ergebnisse für Q_l und E_k für die einzelnen Testvorlagen. Nach der Gleichung 5.1 gibt Q_l den prozentualen Wert der vom entwickelten Programm korrekt identifizierten Textpixel an. E_k gibt den prozentualen Wert für die in der darauffolgenden Texterkennung korrekt erkannten Wörter an. Man kann erwarten, dass diese beiden Werte voneinander abhängen: Eine hohe Anzahl erkannter Textpixel durch das entwickelte Programm sollte zu einer großen Anzahl von erkannten Worten bei der OCR führen. Dieser Zusammenhang zwischen den Größen Q_l und E_k läßt sich mit Hilfe des Korrelationskoeffizienten (Gleichung 5.4) berechnen [21]:

$$r = \frac{\sum_{i=1}^{n}(Q_{l_i} - \bar{Q}_l) * (E_{k_i} * \bar{E}_k)}{(n-1) * s_{Q_l} * s_{E_k}} \,. \tag{5.4}$$

Dabei sind $\bar{Q}_l$ und $\bar{E}_k$ die Mittelwerte und S_{Q_l} und S_{E_k} die Standardabweichungen der beiden Größen Q_l und E_k. Für die Testvorlagen ergibt sich ein Wert von $r = 0,69$ für die beiden korrelierten Variablen Q_l und E_k, so dass die beiden Werte wie erwartet signifikant voneinander abhängen[1].
Betrachtet man die einzelnen Ergebnisse für Q_l und E_k, so stellt man einige Gemeinsamkeiten fest:

- Wie bereits erwähnt führt ein prozentual großer Wert an korrekt erkannten Textpixeln zu einer großen Anzahl von korrekt erkannten Wörtern bei der OCR.

- Bei den Testvorlagen kann man erkennen, dass ein gleichmäßiger Hintergrund im Bereich der enthalten Texte zu einer hohen Anzahl korrekt erkannter Text-

[1] Für weitere Grundlagen zur Statistik sei auf [12] verwiesen.

pixel führt. Beispiele hierfür sind die Testvorlagen Nr. 016, 021, 026, 027 und 030.

- Auch Testvorlagen mit einer hohen Anzahl von Texten wie zum Beispiel die Vorlagen Nummer 004 und 007 stellen bei der Verarbeitung kein Problem dar und erzielen eine hohe Anzahl korrekt erkannter Worte. Wenngleich man auch zum Teil feststellen kann, dass eine hohe Anzahl von Texten verbunden mit einer hohen Anzahl unterschiedlicher Schriftgrößen und Anordnungen innerhalb der Testvorlagen zu geringeren Werten bei der Detektion der Textpixel durch das entwickelte Programm führen kann. Ein Beispiel wäre die Testvorlage Nummer 009.

- Kleine Schriften werden vom Programm gefunden und bei der OCR erkannt. Die Testvorlagen Nummer 019 und 021 enthalten unter anderem Schriften mit einer Höhe im Bereich von 9 bis 11 Pixeln. Bereits bei der visuellen Betrachtung am Fernsehgerät sind diese Schriften schwer lesbar (dies ist sicherlich zum Teil beabsichtigt, da es sich oftmals um Kostenangaben zu gebührenpflichtigen Rufnummern handelt). Dennoch werden diese Schriften vom entwickelten Programm erkannt und bei der Vorlage 021 komplett bei der Vorlage 019 zum Teil durch die OCR in Schrift umgewandelt.

Insgesamt ist eine hohe Streuung der einzelnen Werte für Q_l und E_k zu beobachten. Berechnet man die Standardabweichung für die Erkennungsrate E_k der Wörter nach der Formel 5.5, so erhält man als Ergebnis für $S_{(E_k)}$ circa 22%. Für $S_{(Q_l)}$ erhält man

$$S = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}} \tag{5.5}$$

n: Anzahl der Messwerte,

x_i: Werte der einzelnen Messwerte,

$\bar{x}$: Mittelwert aller Messwerte.

circa 23%. Diese relativ hohe Streuung ist sicherlich bedingt durch die Vielzahl der Parameter die auf die Verarbeitung der Testvorlagen einen Einfluß hat.

Kapitel 6

Zusammenfassung

Ziel der Arbeit ist es, Möglichkeiten für das Erkennen und Extrahieren von Schriften in Videobildern zu finden und diese für Textverarbeitungsprogramme durch Verarbeitung in OCR-Software zugänglich zu machen. Anwendungsfelder sind beispielsweise die automatische Aufzeichnung von Werbetexten oder Texteinblendungen in Nachrichtensendungen. Mit Hilfe eines im Rahmen der Arbeit entwickelten Programms werden die Schriften in hierfür erstellten Testvorlagen die aus unterschiedlichen Fernsehsendungen gewonnen wurden detektiert, zur Verbesserung der Auflösung interpoliert, vom Hintergrund segmentiert und schlußendlich in neuen Bilddateien für die weitergehende Verarbeitung mittels kommerzieller Texterkennungssoftware abgespeichert. Die Verarbeitung wird unter anderem durch unterschiedliche für die Schriften in den Videobildern typische Charakteristika gestützt. Für die Verarbeitung werden die Testvorlagen zuerst in ein Graustufenbild umgewandelt. Die Lokalisierung der Textbereiche geschieht mit Hilfe von Kantenoperatoren und darauffolgender zeilen- und spaltenweiser Projektion der Anzahl der gefundenen Kanten. Die gefundenen Textbereiche werden werden in Vorbereitung auf die später erfolgende Texterkennung in kommerzieller OCR-Software durch bilineare Interpolation in der Auflösung erhöht. Mit Hilfe eines geeigneten, durch das Programm ermittelten Schwellenwertes werden die einzelnen Schriftteile vom Hintergrund getrennt, wobei geeignet hierbei heißt, dass eine möglichst genaue Trennung zwischen der Schrift und dem Hintergrund erreicht werden soll. Helle Schrift vor dunklem Hintergrund wird invertiert. Nach einem letzten Verarbeitungsschritt der einer Re-

duktion von möglichen Fehlern dient, werden die Binärbilder in neuen Bitmapdateien abgespeichert. Diese Binärbilder enthalten nach optimaler Verarbeitung nur noch schwarze Schrift vor weißem Hintergrund. Diese neu erstellten Dateien werden jetzt kommerzieller OCR-Software zugeführt. Schlußendlich erfolgt die Auswertung der Resultate im Hinblick auf die Anzahl der durch das entwickelte Programm erkannten Textpixel in den Testvorlagen und der Anzahl der durch durch die OCR-Software korrekt erkannten Wörter in den Binärbildern.

Kapitel 7

Fazit

Die Medien Text, Video, Fernsehen, Ton und Grafik werden, beispielsweise im Bereich des Internets weiterhin zunehmend miteinander verschmelzen. Sicherlich wird es dadurch auch in zunehmendem Maße von Interesse sein, multimediale Inhalte nach unterschiedlichen Kriterien in Datenbanken oder ähnlichen indexorientierten Strukturen zu erfassen.

Betrachtet man das insgesamt erzielte Resultat von durchschnittlich circa 81% korrekt erkannter Texte in den Testvorlagen, so reicht dies für die in Kapitel eins genannten Anwendungsmöglichkeiten sicherlich in den meisten Fällen aus. Bei der Aufbereitung von erkannten Texten in Videobildern wird es außerdem in vielen Fällen zu einer Verknüpfung mit diversen anderen bekannten Informationen kommen wodurch die extrahierten Texte zum Beispiel innerhalb von Datenbankanwendungen eine zusätzliche Informationsquelle darstellen. Unter dem Aspekt einer ständig steigenden Leistungsfähigkeit moderner Rechnersysteme lassen sich wahrscheinlich noch komplexere und leistungsfähigere als die in dieser Arbeit verwendeten Algorithmen für eine Auswertung der Videobilder finden. Eine Verbesserung der Erkennungsleistung könnte aber auch durch die gezielte Optimierung der einzelnen Verarbeitungsschritte während der Texterkennung auf spezielle Anwendungsfälle beziehungsweise Sendeformate erfolgen. Wenngleich die erzielten Resultate einer relativ großen Schwankungsbreite unterliegen, so sind die erzielten Texterkennungsraten insgesamt für die Testvorlagen sicherlich als recht hoch zu beurteilen.

In allen Fällen wird eine hundertprozentige Erkennungsleistung trozdem nicht zu

erreichen sein, da die Anzahl jener Parameter, die Einfluß auf die Verarbeitung der Videobilder haben hierfür zu groß ist. Für die genannten Anwendungen allerdings wird eine hundertprozentige Erkennungsleistung wahrscheinlich auch nicht benötigt werden.

A. Literaturverzeichnis

[**01**] W. Abmayr: Einführung in die digitale Bildverarbeitung. Teubner Verlag, Stuttgart, 1994.

[**02**] P. Haberäcker: Digitale Bildverarbeitung. Hanser, München, 1991.

[**03**] Microsoft Corporation: http:\\msdn.microsoft.com, Stichwort: Windows GDI Bitmap Structures, Abruf im April 2004.

[**04**] C. Kuhmünch, W. Effelsberg, R. Lienhart: On the detection and recognition of television commercials. In proceedings IEEE Conf. Multimedia and Systems, Ottawa - Canada, Seite 509-516, Juni 1997.

[**05**] M. Seul, L.O'Gorman, M.J. Sammon: Practical Algorithms for Image Analysis - Description, Examples and Code. Cambridge University Press, Cambridge - UK, 2001.

[**06**] D. Lopresti, J. Zhou. Locating and recognizing text in WWW images: Info Retrieval, vol.2, Kluwer Academic Publishers, Murray Hill - USA, Seite 177-206, April 1999.

[**07**] H. Wactlar, M. Christel, Y. Gong, A. Hauptmann: Lessons Learned from Building a Terabyte Digital Video Library, IEEE Computer Society Press, Volume 32, Los Alamitos - USA, Seite 66-73, 1999.

[**08**] R. Lienhart, A. Wernicke: Localizing and Segmenting Text in Images and Videos IEEE Transactions and Systems for Video Technology, Volume 12 No. 4, Los Alamitos - USA, Seite 256-268, April 2002.

[**09**] W. Effelsberg, R. Lienhart: Automatic Text Segmentation and Text Recognition for Video Indexing. ACM/Springer Multimedia Systems Magazine, New York - USA, Seite 3-4, September 1998.

[**10**] Abbyy Software House: http://www.abbyyusa.com, Stichwort: FineReader, Abruf im April 2004.

[**11**] A. Wernicke und R. Lienhart: Text Localization and Text Segmentation in Images, Videos and Web Pages. Informatik Aktuell - Mustererkennung 2000, Springer Verlag, Kiel, September 2000.

[**12**] P. Zamperoni: Methoden der digitalen Bildverarbeitung. Vieweg, Braunschweig / Wiesbaden, 1989.

[**13**] T. Sato, T. Kanade, E. Hughes, M. Smith-Shin'ichi-Satoh: Video OCR: Indexing Digital News by Recognition of Superimposed Caption, Multimedia Systems, Vol. 7, Springer-Verlag New York - USA , Seite 385-395, 1999.

[**14**] H. Li, O. Kia, D. Doermann: Text Enhancement in Digital Video. In Proceedings of SPIE (International Society of Optical Engineering), Document Recognition IV, 1999.

[**15**] S.M. Smith, J.M. Brady: SUSAN - A New Approach to Low Level Image Processing, International Journal of Computer Vision 23, Source Code(Abruf im März 2004): http://www.fmrib.ox.ac.uk/steve/susan/susan21.c , Oxford - England, Mai 1997

[**16**] H.A. Mallot: Sehen und die Verarbeitung visueller Information. Vieweg, Braunschweig / Wiesbaden, 2000.

[**17**] R. Steinbrecher: Bildverarbeitung in der Praxis. Oldenbourg Verlag, München, 1993.

[**18**] B. Jähne: Digitale Bildverarbeitung. 5. Auflage, Springer-Verlag, Berlin / Heidelberg, 2002.

[**19**] P. Haberäcker: Praxis der digitalen Bildverarbeitung und Mustererkennung. Hanser-Verlag, München, 1995.

[**20**] Universität Jena: ICE-Bibliothek, Institut für Informatik Universität Jena, http://pandora.inf.uni-jena.de/ice.html, Abruf im März 2004.

[**21**] P. Zöfel: Statistik für Wirtschaftswissenschaftler. Pearson Studium, München, 2003.

[**22**] U. Breymann: C++ - Eine Einführung. Hanser Verlag, München / Wien, 2001.

[**23**] D. Zimmer, E. Bonz: Objektorientierte Bildverarbeitung - Datenstrukturen und Anwendungen in C++. Hanser Verlag, München / Wien, 1996.

[**24**] D. Louis: C/C++ - Das komplette Programmierwissen für Studium und Job. Markt und Technik, München, 2003.

B. Anhang

B.1 Übersicht über die Testvorlagen

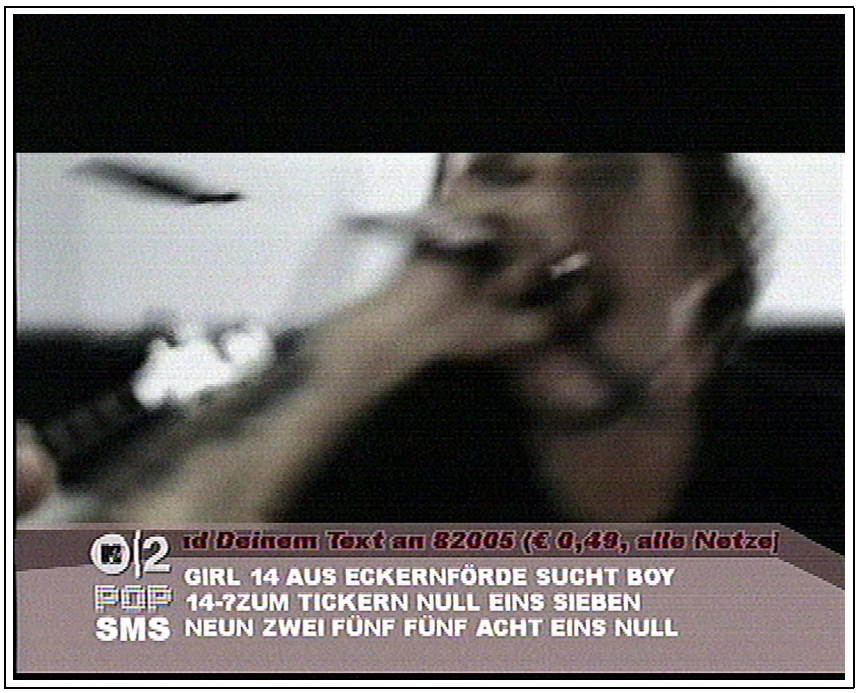


Abbildung B.1: Testvorlage: 001.bmp

Abbildung B.2: Testvorlage: 002.bmp

Abbildung B.3: Testvorlage: 003.bmp

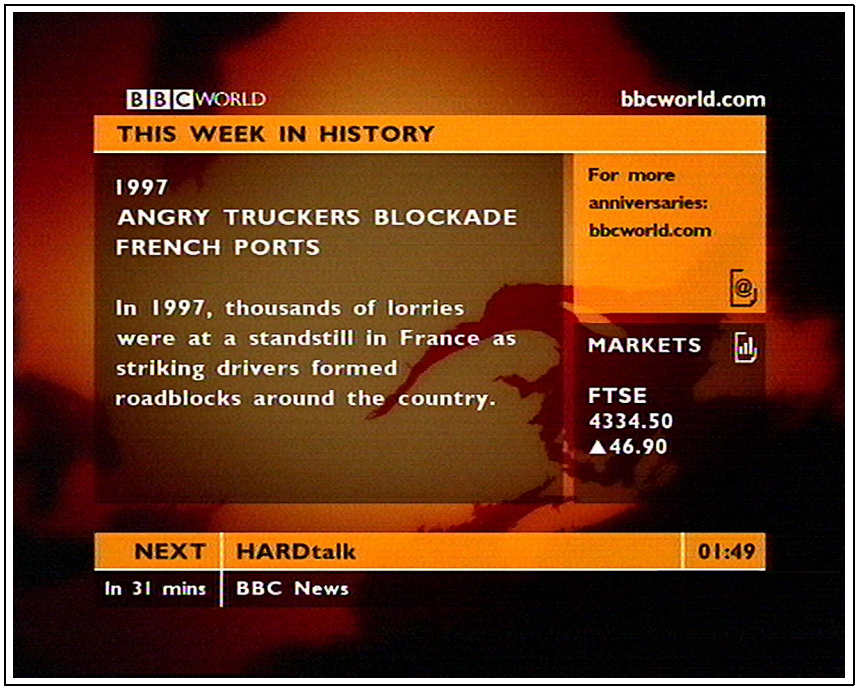


Abbildung B.4: Testvorlage: 004.bmp

Abbildung B.5: Testvorlage: 005.bmp

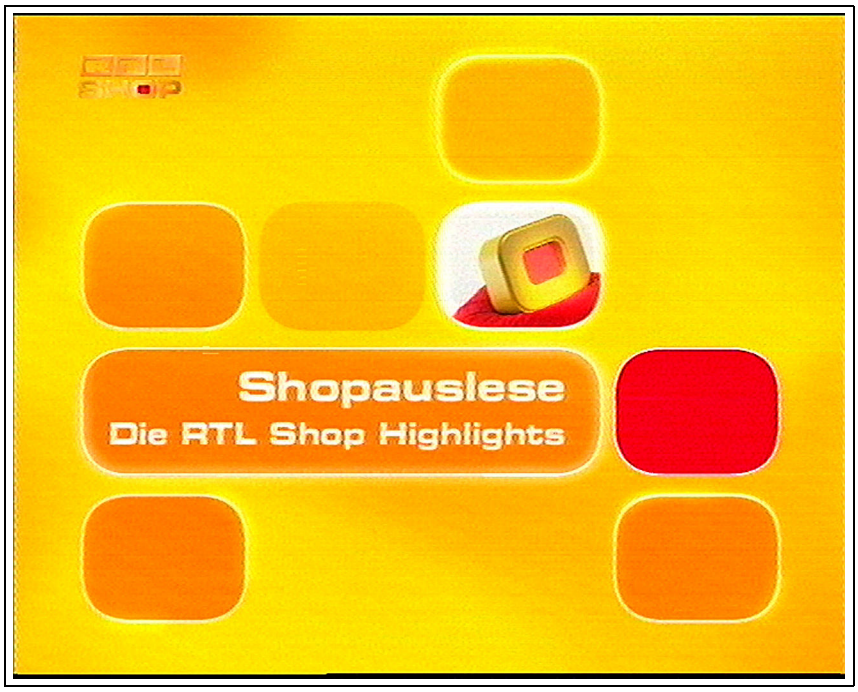


Abbildung B.6: Testvorlage: 006.bmp

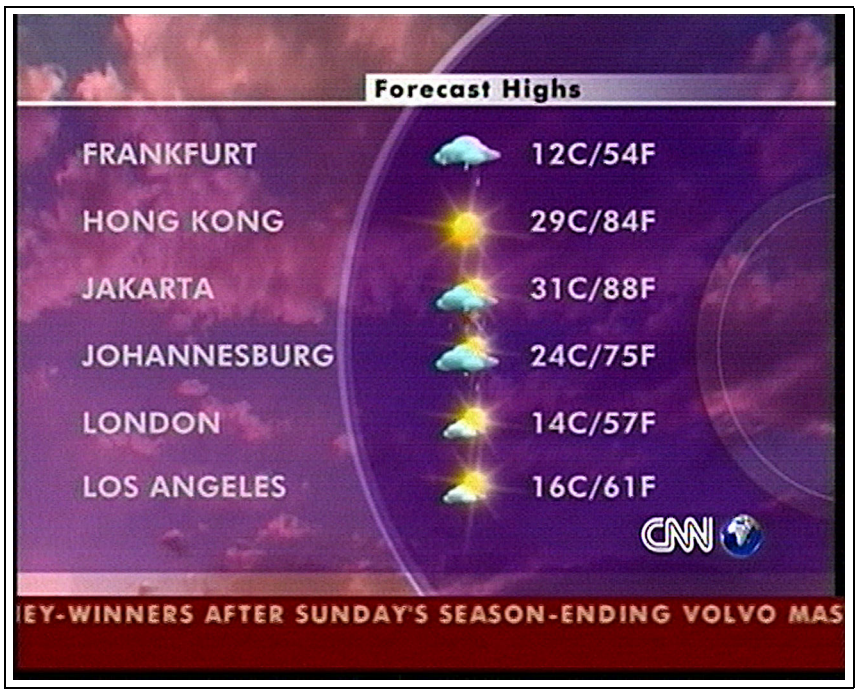


Abbildung B.7: Testvorlage: 007.bmp

Abbildung B.8: Testvorlage: 008.bmp

Abbildung B.9: Testvorlage: 009.bmp

Abbildung B.10: Testvorlage: 010.bmp

Abbildung B.11: Testvorlage: 011.bmp

Abbildung B.12: Testvorlage: 012.bmp

Abbildung B.13: Testvorlage: 013.bmp

Abbildung B.14: Testvorlage: 014.bmp

Abbildung B.15: Testvorlage: 015.bmp

Abbildung B.16: Testvorlage: 016.bmp

Abbildung B.17: Testvorlage: 017.bmp

Abbildung B.18: Testvorlage: 018.bmp

Abbildung B.19: Testvorlage: 019.bmp

Abbildung B.20: Testvorlage: 020.bmp

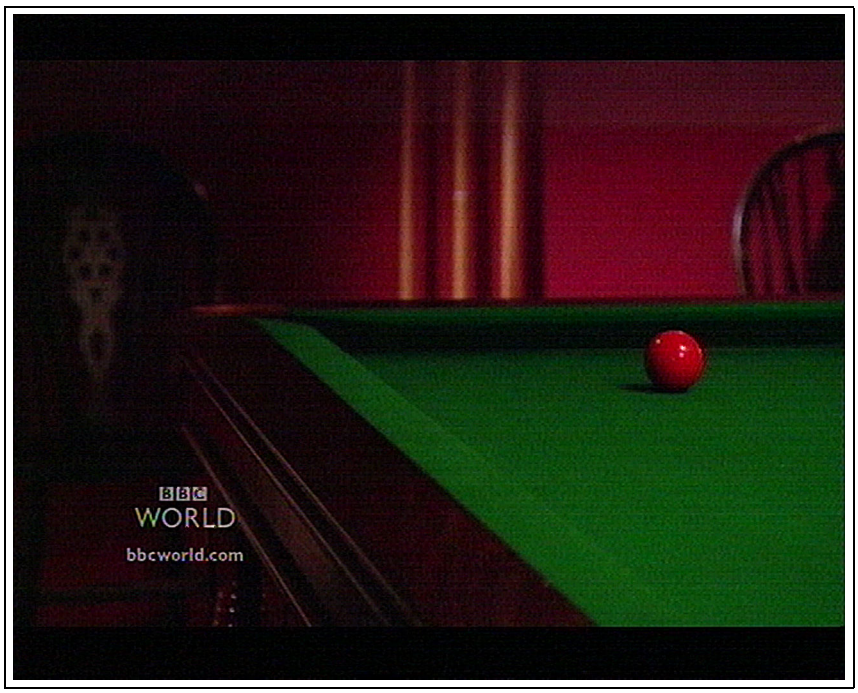


Abbildung B.21: Testvorlage: 021.bmp

Abbildung B.22: Testvorlage: 022.bmp

Abbildung B.23: Testvorlage: 023.bmp

Abbildung B.24: Testvorlage: 024.bmp

Abbildung B.25: Testvorlage: 025.bmp

Abbildung B.26: Testvorlage: 026.bmp

Abbildung B.27: Testvorlage: 027.bmp

Abbildung B.28: Testvorlage: 028.bmp

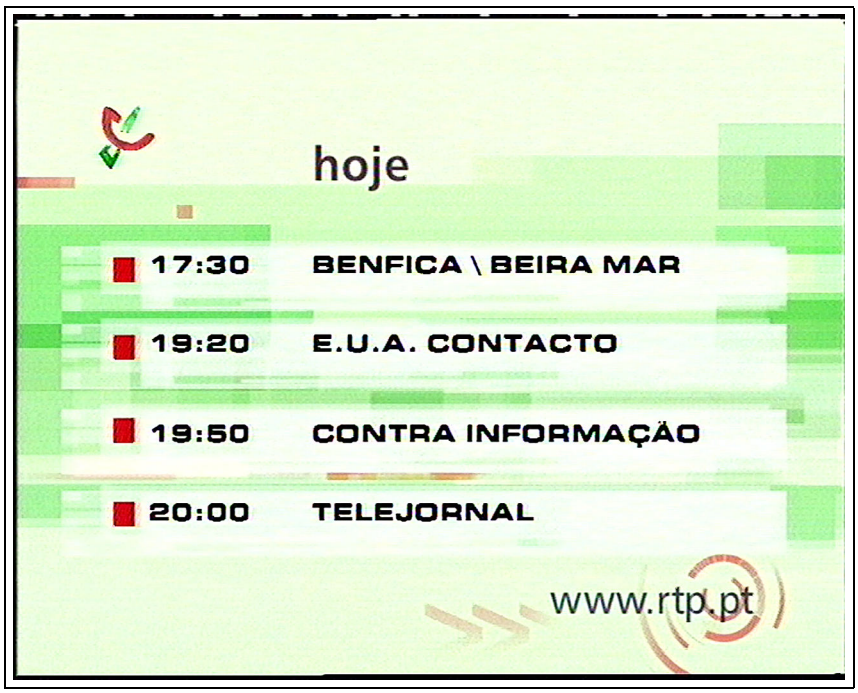


Abbildung B.29: Testvorlage: 029.bmp

Abbildung B.30: Testvorlage: 030.bmp

Eidesstattliche Erklärung

Ich versichere hiermit, die vorgelegte Arbeit in dem gemeldeten Zeitraum ohne fremde Hilfe verfaßt und mich keiner anderen als der angegebenen Hilfsmittel und Quellen bedient zu haben.

Köln, den 24.05.2004

Axel Näther

Sperrvermerk

Die vorgelegte Arbeit unterliegt keinem Sperrvermerk.

Weitergabeerklärung

Ich erkläre hiermit mein Einverständnis, dass das vorliegende Exemplar meiner Diplomarbeit oder eine Kopie hiervon für wissenschaftliche Zwecke verwendet werden darf.

Köln, den 24.05.2004

Axel Näther